Máster Oficial en Sistemas Multimedia

Departamento de Teoría de la Señal, Telemática y Comunicaciones

Universidad de Granada

Tesis doctoral:

Una aproximación Publicación/Subscripción Centrada en Datos para la Provisión de Servicios Multimedia

Realizado por:
Javier Povedano Molina

Dirigido por:
Dr. D. Juan Manuel López Soler

Editor: Editorial de la Universidad de Granada
Autor: Javier Povedano Molina
D.L.: GR 361-2014
ISBN: 978-84-9028-766-8

Official Graduate Program in Multimedia Systems

Department of Signal Theory, Telematics and Communications

University of Granada

Ph.D. Thesis dissertation:

A Data-Centric Publish/Subscribe Approach for Multimedia Service Provisioning

Written by:
Javier Povedano Molina

Supervised by:
Juan Manuel López Soler, Ph.D.

El doctorando D. Javier Povedano Molina y el director de la tesis Dr. D. Juan Manuel Lopez Soler, Profesor Titular de Universidad del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada,

Garantizamos, al firmar esta tesis doctoral,

que el trabajo ha sido realizado por el doctorando bajo la dirección del director de la tesis y hasta donde mi conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, a 21 de Mayo de 2013

Director de la tesis:	Doctorando:
Fdo: Juan Manuel Lopez Soler	Fdo: Javier Povedano Molina

Agradecimientos

En primer lugar quiero agradecer a mi director, Juanma por haberme guiado en todo este proceso, ayudándome a formarme en este complicado mundo de la investigación.

A mis compañeros del grupo de investigación Wireless and Multimedia Networking Lab (WMNL), por todos los ratos y experiencias vividas durante estos años en la Universidad, especialmente a Juanjo y José María, que con paciencia y humor han hecho más llevaderas las largas sesiones de experimentos.

Quiero agradecer también a todos mis compañeros del Departamento de Teoría de la Señal, Telemática y Comunicaciones por haberme hecho sentir totalmente integrado, especialmente por esas cervecitas de los viernes donde nos olvidábamos por un rato de los problemas del día a día entre tapas y sesiones de chistes.

Al Prof. Antonio Corradi y Luca, de la Universidad de Bolonia por haberme acogido y permitido vivir una experiencia inolvidable en su preciosa ciudad.

Agradecer también a mis compañeros del *CITIC* por las vivencias durante estos años, y que me han enseñado a tomarme con humor y chascarrillos todas las vicisitudes que se han interpuesto en nuestra formación como investigadores. A Rosa y Natalia, mis compañeras de *tupperware* por las conversaciones en las que arreglábamos el mundo a la hora del almuerzo. A mis compañeros de mi laboratorio y el contiguo por esos desayunos y bromas que te hacían sacar una sonrisa aunque los experimentos no salieran como querías. A la máquina de café, por su apoyo incondicional ;).

A todos los *Bohemios*, que han estado ahí durante todo este tiempo, apoyándome y animándome a pesar de mirarme raro cuando intentaba explicarles de qué iba mi tesis.

A mis padres, hermano y familia, por haberme apoyado en todo momento en cada una de mis decisiones y haber hecho posible encontrarme en este punto.

Finalmente, agradecérselo a Laura, que con infinita paciencia me ha acompañado y animado en cada uno de mis altibajos durante el desarrollo de esta Tesis.

A todos, GRACIAS.

Abstract

Recent advances in the performance of computers and mobile devices, as well as improvements in networks and communication infrastructures are making possible that a plethora of real time generated multimedia contents are easily and universally available. This is even possible in scenarios where multiple heterogeneous devices access to the multimedia content. For that, it is necessary to have a powerful infrastructure behind the scenes aimed at adapting, transforming and distributing the multimedia contents according to the end user requirements. Thanks to this infrastructure, scenarios such as video surveillance, border control and collaborative environments –among many others– have increased both their functionalities and the quality of the provided service as well.

Traditionally, such kind of systems have been designed by using a centralized approach that relies on the server/client paradigm. In this paradigm, a central server is responsible of processing, transforming and disseminating the multimedia streams to interested clients, under the request/reply interaction pattern. In spite of being highly adopted, this interaction model raises some well-known problems, such as the discovery of the multimedia content and services available, and the limited scalability of the system, due to the server potentially becomes a bottleneck and a central point of failure. These issues, shared by almost every centralized system, become even worse in the case of multimedia applications, as they are highly resource-demanding and delay sensitive kind of data.

To overcome these issues, this Thesis proposes an architecture for distributing and providing multimedia services. The proposed framework features robustness, efficiency, scalability and flexibility properties. Our approach is essentially based on a publish/subscribe data-centric interaction model.

In this paradigm, the temporal and spatial couplings –always present in clien-

t/server interactions– between data producers and consumers are removed. Added to that, the data-centricity approach has a double benefit. First, it additionally decouples the application level, in such a way that the deployment of new services can dynamically evolve in time with no interruptions, and secondly, it means that data are not opaque to the middleware. This feature reduces the application logic complexity, and improves the performance of the system.

Related to that, for maximizing the throughput and for detecting anomalous behaviors and overloading, in any distributed system and, particularly in those providing multimedia services, to have a monitoring framework is a must. The monitoring issue is gaining momentum due to the implosion of cloud computing systems. Although a number of significant approaches have been proposed none of them solve this problem as our approach does.

In this respect, this Thesis is aimed at solving two main aspects from a common perspective: first, the building and compositing of multimedia services, and, second the resource monitoring in highly dynamic cloud computing environment –such as distributed multimedia cloud processing–. For both problems we adopt a common publish/subscribe data-centric approach.

More precisely, we propose Extensible Multimedia Distribution Service (EMDS) a component-based distributed system aimed at building and composing multimedia services. We explain the Extensible Multimedia Distribution Service (EMDS) design as well as its implementation details. After the conducted evaluation, the results show that EMDS performances efficiently (by reducing the consumed bandwidth and the latency compared to other reference systems). Besides of that, the provided discussion justifies the EMDS robustness, scalability and, flexibility.

Additionally, this Thesis also proposes Distributed Architecture for Resource manaGement and mOnitoring in cloudS (DARGOS) a monitoring system that takes advantage of the data-centric publish/subscribe paradigm in cloud computing environments. Similarly to the EMDS explanation, we provide the most relevant aspects of DARGOS design as well as the details of a prototype implementation. DARGOS prototype has been exhaustively evaluated, in different deployments ranging from local and intermediate deployments to massive ones which emulate a real data-center deployed in remote sites connected by Wide Area Networks (WANs). The results quantitatively show the benefits of the proposed system and more precisely, we demonstrate that under the same conditions DARGOS favorably compares to other widely used monitoring systems.

Resumen

Las mejoras recientes en las prestaciones de los computadores y dispositivos móviles, así como en las redes e infraestructuras de comunicación, han hecho posible que diferentes contenidos multimedia generados en tiempo real puedan estar al alcance de nuestra mano en todo momento. En algunos casos, esto es incluso posible a pesar de la gran heterogeneidad de los dispositivos de acceso al contenido multimedia. Para ello, es necesario disponer de una potente infraestructura que se encargue de adaptar, transformar y distribuir los flujos multimedia para satisfacer los requerimientos del usuario final. Gracias a esta infraestructura, escenarios como la vídeo-vigilancia, el control de fronteras o los entornos colaborativos –entre otros muchos– han incrementado sus funcionalidades y la calidad del servicio prestado.

Clásicamente, este tipo de sistemas se han diseñado adoptando una aproximación centralizada basada en el paradigma de interacción cliente/servidor. En este paradigma, un servidor central es el encargado de procesar, transformar y difundir bajo un esquema de petición/respuesta los flujos multimedia a los clientes interesados. A pesar de su elevada penetración, este modelo de provisión origina varios problemas entre los que destacan la rigidez en el descubrimiento de los recursos y servicios disponibles, así como la baja escalabilidad de los correspondientes sistemas que pueden convertirse en cuello de botella. Además es bien conocida la debilidad de los sistemas centralizados por su dependencia de un único punto central. Estos problemas, que son comunes a todos los sistemas centralizados, se agravan aún más en el caso de aplicaciones multimedia ya que por su propia naturaleza son muy demandantes de recursos, siendo a su vez muy sensibles a retardos.

Para paliar estos problemas esta Tesis propone una arquitectura para la provisión de servicios multimedia caracterizada por su robustez, eficacia, escalabilidad y flexibilidad. En el diseño propuesto es de suma importancia la adopción de un

modelo de interacción publicación/subscripción centrado en datos.

En este paradigma el acoplamiento espacial y temporal -característico del modelo cliente/servidor- entre productores y consumidores de información es eliminado, haciendo posible el desarrollo de nuevos entornos de provisión y despliegue de servicios. Además el carácter centrado en datos (*data-centric*) tiene un doble beneficio. Por un lado, permite un desacoplamiento adicional a nivel de aplicación, tal que el despliegue de servicios puede evolucionar en el tiempo y sin interrupciones y por otro lado, significa que el *middleware* no es indiferente a los datos, al contrario, lo que le permite asumir responsabilidades que en definitiva simplifican la lógica de la aplicación y consiguen un mejor uso de los recursos implicados.

Relacionado con lo anterior, para conseguir el máximo rendimiento o para detectar posibles comportamientos anómalos o sobrecargas, en todo sistema distribuido -y en particular en aquellos para la provisión de servicios multimedia- es imprescindible disponer de un entorno de monitorización de recursos. El problema de la monitorización recientemente ha retomado un impulso creciente debido a la generalización de los sistemas de computación en la nube (*cloud computing*) y aunque es posible encontrar varios sistemas de referencia en este contexto, pocas veces se ha afrontado este problema adoptando una aproximación como la que se propone en esta Tesis.

En este sentido el trabajo realizado se ha centrado en resolver dos problemas relacionados desde una perspectiva común: de un lado la provisión de servicios y distribución de multimedia y por otro, la monitorización de los recursos en un entorno de *cloud computing* altamente cambiante –como puede ser un entorno multimedia– adoptando un paradigma común de publicación/subscripción centrado en datos.

Más concretamente, en este trabajo se propone Extensible Multimedia Distribution Service (EMDS) un sistema basado en componentes para la construcción y composición de servicios multimedia distribuidos. Además de explicar el diseño y la implementación realizada, se demuestra experimentalmente cómo EMDS verifica los requisitos de eficacia (reduciendo el ancho de banda consumido y el retardo frente a otras sistemas) tanto en la provisión de servicios como en la distribución de contenidos. Además en el estudio y discusión realizados se justifica su robustez, escalabilidad y flexibilidad.

Adicionalmente se propone DARGOS, un sistema de monitorización de recursos que aprovecha las ventajas del paradigma de publicación/suscripción centrada en datos en entornos de *cloud computing*. Al igual que en EMDS, para DARGOS se explican los aspectos más relevantes de su diseño así como la implementación realizada. El prototipo se ha evaluado exhaustivamente, en diferentes entornos que van desde un entorno local pasando por despliegues intermedios hasta despliegues masivos que emulan a un *data-center* real desplegado en sedes remotas a través de WAN. Los resultados obtenidos demuestran los beneficios del sistema propuesto y

en particular evidencian que DARGOS en igualdad de condiciones se compara favorablemente con otros sistemas de monitorización ampliamente difundidos en la bibliografía.

Índice general

1. Introduction **1**

1.1. Motivation 2

1.1.1. Distributed multimedia processing 3

1.1.2. Distributed system monitoring 5

1.2. Goals 5

1.3. Solution overview 6

1.4. Main contributions 6

1.4.1. Works psublished 7

1.5. Document structure 8

1. Introducción **11**

1.1. Motivación 11

1.1.1. Procesamiento multimedia distribuido 14

1.1.2. Monitorización de sistemas distribuidos 16

1.2. Objetivos 16

1.3. Resumen de la solución propuesta 17

1.4. Contribuciones de esta tesis 17

1.4.1. Trabajos publicados 18
1.5. Estructura del documento 20

2. Trabajo Relacionado **21**
2.1. Publicación/subscripción 21
2.1.1. *Message Queue Telemetry Transport* (MQTT) 23
2.1.2. Java Message Service (JMS) 25
2.1.3. Advanced Message Queuing Protocol (AMQP) 27
2.1.4. Data Distribution Service (DDS) 29
2.1.4.1. Data Centric Publish Subscribe (DCPS) 30
2.1.4.2. Data Local Reconstruction Layer (DLRL) 31
2.1.4.3. Real-Time Publish Subscribe (RTPS) 33
2.1.4.4. Características relevantes de DDS 33
2.1.5. Otros sistemas de publicación/subscripción 34
2.1.6. Comparativa de estándares 36
2.2. Servicios multimedia 38
2.2.1. Perspectiva histórica 40
2.2.2. Multimedia y publicación/subscripción 42
2.2.2.1. CORBA/NaradaBrokering 42
2.2.3. Sistemas *peer to peer* 44
2.2.4. Sistemas de *Cloud computing* 45
2.2.5. Otros sistemas relevantes relacionados. 46
2.2.6. Resumen. 47
2.3. Monitorización de recursos en entornos distribuidos 48
2.3.1. Antecedentes históricos 48
2.3.2. Clasificación de los sistemas de monitorización 49
2.3.3. Nagios 53

2.3.4. Ganglia 56
2.3.5. Lattice 57
2.3.6. Otros sistemas de monitorización relevantes 58
2.3.7. Comparación cualitativa entre sistemas de monitorización . . 59

3. Una arquitectura extensible para servicios multimedia 61
3.1. Requisitos 62
3.2. Diseño 64
3.2.1. Módulo de descubrimiento 65
3.2.2. Módulo de codificación 68
3.2.3. Módulo de comunicación 70
3.2.4. Módulo de filtrado 73
3.2.5. Módulo de seguridad 74
3.2.6. Diagrama de clases 76
3.3. Implementación 78
3.4. Construcción de servicios EMDS 80
3.4.1. Transformaciones 81
3.4.2. Análisis 82
3.4.3. Procesamiento 83
3.4.4. Composición 84
3.4.5. Pasarelas 84
3.4.6. Realidad aumentada 85
3.4.7. Respaldo 86
3.5. Casos de uso y servicios multimedia EMDS en el *Cloud* 87
3.6. Validación de la arquitectura 89
3.6.1. Descripción del escenario experimental 90
3.6.2. Estudio del retardo punto a punto 92

3.6.3. Eficacia en el despliegue de servicios. Impacto en el retardo . . 92
3.6.4. Impacto del uso de comunicaciones fiables 94
3.6.5. Evaluación del impacto de la fragmentación 95
3.6.6. Evaluación del impacto de la seguridad 97
3.7. Conclusiones 98

4. Monitorización de recursos en entornos distribuidos 101
4.1. Requerimientos 102
4.2. DARGOS 104
4.2.1. Diseño 105
4.2.1.1. Modelo de datos 106
4.2.1.2. Diagrama de clases 107
4.2.2. Agentes DARGOS 108
4.2.3. Componentes principales 110
4.2.3.1. Módulo de comunicación 110
4.2.3.2. Módulo de recolección de datos 118
4.2.3.3. Módulo de control 120
4.2.3.4. Módulo de consulta 121
4.2.3.5. Módulo de seguridad 123
4.3. Implementación 126
4.3.1. Seguridad 127
4.3.2. Implementación de tópicos DDS 129
4.4. Evaluación de DARGOS 130
4.4.1. Descripción del entorno experimental 131
4.4.2. Evaluación del tráfico generado 134
4.4.2.1. Impacto del número de sensores y subscriptores . . . 135
4.4.2.2. Comparativa con *Nagios* y *Lattice* 136

4.4.3. Impacto de la seguridad 140
4.4.4. Escalabilidad del sistema 141
4.5. Conclusiones 146

5. Conclusiones **151**
5.1. Conclusiones 151
5.2. Trabajo futuro 154

5. Conclusions **155**
5.1. Conclusions 155
5.2. Future work 158

Bibliografía **159**

Índice de figuras

2.1. Paradigma de publicación/subscripción. 22
2.2. Arquitectura JMS. 26
2.3. Arquitectura AMQP. 27
2.4. Entidades DDS y sus interrelaciones. 30
2.5. H.323: estándares y protocolos asociados 41
2.6. Publicación de audio/vídeo basada en *NaradaBrokering*. 43
2.7. Red *peer to peer* para la distribución de contenido multimedia. 44
2.8. Clasificación de los sistemas de monitorización 50
2.9. Centralizado vs. distribuido 51
2.10. Pull vs. Push 52
2.11. Captura de Nagios 54
2.12. Extensiones de Nagios: Nagios Remote Plugin Extension (NRPE) y Nagios Serviche Check Acceptor (NSCA). 55
2.13. Arquitectura de *Ganglia*. 57
2.14. Arquitectura de Lattice. 58

3.1. Módulos de EMDS 64

3.2. Codificación de la señal de vídeo: Group Of Pictures (GOP). 69
3.3. Jerarquía de clases de EMDS. 76
3.4. Diagrama de secuencia: Publicación de un flujo de video 78
3.5. Servicio EMDS: Transformación de flujos. 82
3.6. Servicio EMDS: Análisis de flujos 83
3.7. Servicio EMDS: Procesamiento de flujos. 83
3.8. Servicio EMDS: Composición de flujos. 84
3.9. Servicio EMDS: Pasarelas 85
3.10. Servicio EMDS: Realidad aumentada. 86
3.11. Servicio EMDS: Respaldo de flujos. 86
3.12. Escenario de vídeo-vigilancia basado en EMDS. 87
3.13. Escenario de vídeo-vigilancia basado en EMDS: nuevos servicios. . . . 88
3.14. Entorno experimental de evaluación. 91
3.15. Retardo medio. 92
3.16. *Jitter* medio. 93
3.17. Retardo de extremo a extremo. 94
3.18. Retardo al proporcionar un servicio. 95
3.19. Retardo de extremo a extremo: *best-effort* vs. *reliable* 96
3.20. Fragmentación vs. tráfico generado. 97
3.21. Retardo para diferentes tamaños de fragmento: secuencia *video1* 3.21a y *video2* 3.21b. 98
3.22. Impacto del cifrado en el retardo. 99

4.1. Diagrama de clases más relevantes de DARGOS. 108
4.2. NMA y NSA. 109
4.3. Módulos DARGOS. 111
4.4. Módulo de Comunicación. 115
4.5. Gestión de zonas. 116

4.6. Políticas de actualización: Notificación periódica *vs.* basada en eventos. 117
4.7. Módulo de recolección de datos. 119
4.8. Módulo de control. 121
4.9. Módulo de consulta. 123
4.10. Seguridad (autenticación e integridad) en DARGOS. 127
4.11. *Testbeds* 1 y 2. 132
4.12. *Testbed* 3. 133
4.13. *Testbed 4*. 133
4.14. Tráfico generado en DARGOS en función del número de sensores. . . 134
4.15. Tráfico generado en DARGOS en función del número de subscriptores (Node Supervisor Agents (NSAs)). 135
4.16. Comparativa del impacto del número de publicadores en *DARGOS*, *Lattice* y *Nagios*. 138
4.17. Comparativa del impacto del número de subscriptores en DARGOS y *Lattice* para 30 publicadores. 140
4.18. Comparación throughput: DARGOS *vs.* Lattice. 142
4.19. Comparación DARGOS vs. *Ganglia* en WAN: tamaño de mensajes . . . 149
4.20. Comparación DARGOS vs. *Ganglia* en WAN: tráfico total generado . . 149
4.21. Comparación DARGOS vs. *Ganglia* en WAN: ancho de banda 150

Índice de tablas

2.1. Políticas de calidad de servicio definidas por el estándar Data Distribution Service (DDS). 32

2.2. Comparativa entre especificaciones Publicación/Subscripción. 37

2.3. Comparación directa entre sistemas de monitorización de recursos . . 60

3.1. Software empleado para la experimentación. 90

3.2. Vídeos utilizados en la experimentación. 90

3.3. Tamaños de paquete de los vídeos empleados. 91

4.1. Políticas de calidad de servicio (Quality of Service (QoS)) aplicables a la monitorización de recursos en entornos distribuidos. 114

4.2. REST API. 125

4.3. Características hardware/software e identificación de *hosts*. 130

4.4. Versiones software empleadas en la experimentación. 131

4.5. Recursos monitorizados en los experimentos 131

4.6. Número de *hosts* para el *Testbed* 4. 134

4.7. Evaluación cualitativa de DARGOS, *Lattice* y *Nagios*. 137

4.8. Impacto del módulo de seguridad. 141

4.9. Tráfico total generado (Bytes/segundo) por DARGOS, Lattice y Ganglia y los respectivos porcentajes de mejora de DARGOS. 143

4.10. *Round trip time* entre el recolector de datos y los *data centers* 144

Lista de Acrónimos

AMQP Advanced Message Queuing Protocol

API Application Programming Interface

ATM Asynchronous Transfer Mode

CCTV Closed Circuit TeleVision

CDR Common Data Representation

CGI Common Gateway Interface

CORBA Common Object Request Broker Architecture

CPU Central Processing Unit

CSV Comma Separated Values

DARGOS Distributed Architecture for Resource manaGement and mOnitoring in cloudS

DCPS Data Centric Publish Subscribe

DDS Data Distribution Service

DHT Distributed Hash Table

DLRL Data Local Reconstruction Layer

DTD Document Type Definition

DTLS Datagram Transport Layer Security

EMDS Extensible Multimedia Distribution Service

EOS End Of Stream

GOP Group Of Pictures

GRIM Grid Resource Information Monitoring

HMAC Hash-based Message Authentication Code

HMP Host Monitoring Protocol

HPC High Performance Computing

HTTP HyperText Transfer Protocol

IETF Internet Engineering Task Force

IP Internet Protocol

ITU-T International Telecommunication Union - Telecommunication Standardization Sector

JCR Journal Citation Reports

JMS Java Message Service

JSON JavaScript Object Notation

LAN Local Area Network

M2M *Machine to Machine*

MCU Multipoint Control Unit

MOM Message Oriented Middleware

MQTT *Message Queue Telemetry Transport*

NAT Network Address Translation

NMA Node Monitoring Agent

NMS Network Management System

NRPE Nagios Remote Plugin Extension

NSA Node Supervisor Agent

NSCA Nagios Serviche Check Acceptor

NTP Network Time Protocol

OHS Open Hypermedia System

OMG Object Management Group

P2P Peer to peer

PIP Picture in Picture

PKI Private Key Infrastructure

PPS Picture Parameter Set

PSIRP Publish Subscribe Internet Routing Paradigm

PTP Precision Time Protocol

QoS Quality of Service

REST REpresentational State Transfer

RFC Request For Proposal

RSS Really Simple Sindication

RTP Real-Time Protocol

RTPS Real-Time Publish Subscribe

RTSP Real-Time Streaming Protocol

SAR Search and Rescue

SDP Simple Discovery Protocol

SIP Session Initiation Protocol

SNMP Simple Network Management Protocol

SOAP Simple Object Access Protocol

SP-II Internet Stream Protocol (version 2)

SPS Sequence Parameter Set

SSH Secure SHell

SVC Scalable Video Coding

TCP Transmission Control Protocol

TLS Transport Layer Security

TOAST Toolkit for Open Adaptive Streaming Technologies

UAV Unmanned Aerial Vehicles

UDP User Datagram Protocol

URL Uniform Resource Locator

UUID Universally Unique IDentifier

VoIP Voice over IP

WAN Wide Area Network

WSDL Web Services Description Language

WSN Wireless Sensor Networks

WWW World Wide Web

XDR eXternal Data Representation

XML eXtensible Markup Language

XMPP eXtensible Mesasaging and Presence Protocol

1

Introduction

The world around us can be represented using multimedia streams. All the information that we see and hear can be captured and transmitted by digital devices located in different geographical locations. However, recent technological advances occurred in the last decade made possible to build more advanced systems that are able to process such information and make it available to multiple devices ranging from personal computers to mobile devices.

One of the fields that has evolved more in this direction is the developing of systems capable of processing such multimedia streams to extract information and adapt them to the devices that are present in our daily life. For instance, it is possible to record a video and make it available to multiple users in real time, no matter what kind of device they are using. This is possible to the processing happening behind the scenes, such dedicated services that transform video streams into different resolutions and formats.

Traditionally, such services were carried by central servers that had the responsibility of performing all the processing needed to carry out such tasks. However, after years it has been proved that such centralized model based on request/reply patterns show multiple weaknesses, such scalability issues and the need to determine the physical location of such services.

Recently, new paradigms aimed to remove the dependency between the contents and the location of the server providing such contents have emerged. For instance, in "Architectural Principles of the Internet" [23], Section 4.1, such statement was made:

– "In general, user applications should use names rather than addresses."

According to this statement, users should not access to resources using the phy-

sical location of the device hosting it, instead, it should be possible to access them using their name. In the case of multimedia contents, there are multiple advantages on using this paradigm shift:

Discovery The discovery and identification of the available multimedia contents is easier as they are no longer associated to a phisical address. Hence, it is possible to access to resources based in their content.

Mobility The contents can be accessed no matter *where* they are originated, thus making it possible that services can be cloned, migrated or moved in real time in a transparent manner for the user.

Extensibility With the disassociation between content and addressing, it is easier to integrate new services that consume the available contents to generate new contents that can be consumed by other services.

One of the new paradigms that are gaining momentum recently is the publish/subscribe paradigm. In this approach, data consumers (subscribers) say *what* information they want to receive, whereas information producer peers (publishers) are responsible to transmitting appropriate data to interested peers. Recent works in processing and dissemination of multimedia content, show the applicability and convenience of this new paradigm [22, 43, 97, 106, 183, 202].

Within the publish/subscribe systems available, there is a type known as data-centric publish/subscribe that allows to apply a new approach in content distribution. In this Thesis, we are going to study the advantages given by this new paradigm in the developing of distributed real-time multimedia systems.

1.1. Motivation

Within the different existing types of publish/subscribe approach, there is one that is gaining momentum recently: the data-centric publish/subscribe. In this approach, the format of the data and their contents define the data flow between publishers and subscribers. Currently, the Object Management Group (OMG) specification DDS is considered as the main standard in this kind of communications.

DDS has been recently adopted as the communications core in many critical systems around the world, ranging from financial environments to tactical warfare environments [152, 153]. All of these scenarios share some common characteristics, being the most important:

- They are composed by multiple data producers and consumers geographically distributed, reaching in some scenarios thousands of them.
- Some of these systems are continously subject to changes and evolutions due to the appear of new requirements or participants.
- Data distribution is carried in real time, being critical to minimize the delays in data transmission.
- Usually, they work with big data volumes, flowing continously between data producers and consumers.

These characteristics fit with the desirable for a distributed multimedia processing system. In this kind of scenarios there are multiple data processing nodes consuming streams from geographically dispersed to produce new streams. Such multimedia streams, usually are high volume and continuous streams very sensitive to anomalous network behaviors such as delays.

In this Thesis we propose a study on the benefits reported on using a data centric approach in the case of distribute and process multimedia content along with the resource monitoring needed for this task.

1.1.1. Distributed multimedia processing

Distributed multimedia processing makes possible building large systems aimed to transform streams from different sources into derived streams or even analyzing them to extract information from its content.

Based on this description, it is possible to find multiple systems in real life that fit it. Detailed below there are some scenarios where processing multimedia streams is very important:

Unmanned Aerial Vehicles (UAV) are aircrafts without a human pilot aboard that are aimed to gather information by air. They are commonly used in military applications such as reconnaissance or attack flights, but they are increasing their presence in civil applications such as fire detection or Search and Rescue (SAR) [66]. Information captured by the aircrafts is used to take decisions by personnel on the ground. The main issue concerning in UAV is that usually have limited resources. Hence, it is necessary to transmit the data gathered to multiple decision taking points in an efficient manner. In such scenarios it is usual that the information is visualized and processed by multiple supervisors to take appropriate decisions. One example of such processing scenario is the use of algorithms for object tracking or anomaly detection by air.

Video Surveillance Video surveillance systems or Closed Circuit TeleVision (CCTV) consist in multiple video cameras deployed to supervise an environment or event. Originally, these cameras were connected directly to displays, but the evolution of electronic devices allowed the interconnection with other devices such as computers or mobile devices used by security personnel. The functionalities that should be covered by these systems are the visualization on devices with heterogeneous capabilities (resolution, bandwidth,...), storage of audiovisual data and the detection and notification of anomalies and suspicious behaviors [21]. This functionality is covered by multiple services deployed in the system that are responsible of discovering the multiple video streams being captured by different cameras. An example of such application is border control or traffic control.

Sports events In certain sport events, it is interesting to integrate the audiovisual information with other information coming from other sensors. An example of this scenario can be seen in a Formula 1 race. In this scenario there are many cars equipped with cameras and sensors that gather various telemetry data, such as the speed of the car, time per lap, or car throttle. Some of this data should be overlayed in real time with the video streams captured by the cameras. Such combined streams can be used for visualization purposes by spectators or to be studied by engineers.

Collaborative environments Another environment where distributed stream processing is interesting is the remote collaboration or teleconference. In this kind of scenarios, many users are connected with microphones and web cameras in a virtual room where they also can share videos or virtual whiteboards [68]. One of the problems that should face such systems is that as they allow multiple users at the same time, they are very resource demanding. This can be even worse in scenarios where devices with heterogeneous capabilities are used at the same time. One solution to these problems, is to use services that combine multiple multimedia streams in one stream that contain all the relevant information: for instance, mixing different audio streams and layout all the videos in a grid. Another solution is to deploy services that create low resolution copies of the multimedia streams.

All the previous scenarios share a common trait: apart from multimedia producers and consumers, they require the deployment of multimedia services that are responsible of perform the processing and transformation of the streams. Usually, such services are deployed using a centralized approach that: should be designed in an *ad hoc* fashion, and, require an important amount of human intervention when they have to be extended with new functionalities or media sources.

For this reason, we propose in this Thesis the use of a data-centric publish/subscribe approach to solve the problems in such centralized services.

1.1.2. Distributed system monitoring

One of the main issues that have to be coped in distributed systems is the control over the status and usage of the resources that are present (Central Processing Unit (CPU), memory, network usage,...). This is more evident in scenarios where huge computational loads changing over the time happen, such as in the case of multimedia stream processing.

Distributed monitoring systems are usually built using a central server responsible of gathering the resource usage statistics using explicit requests to monitoring nodes. However, apart from the well known problems present in centralized systems, other problems have to be faced:

Heterogeneity There are multiple kinds of applications that can be interested in monitoring information, each of them with their own update policies requirements. Whereas interactive applications such as dashboard panels need to receive updates with a frequency low enough to be assimilated by human operators, other applications such as load balancers, need more frequent updates, specially if they need it for taking decisions.

Staticity Many centralized monitoring systems, are designed to manage environments with resources that no vary over the time. Adding new monitoring nodes require a configuration process to allow the server and the client to know each other.

The arrival of new computation paradigms such as Cloud Computing, where resources can be added or removed dynamically on demand, became some of these centralized servers obsolete. For this reason, it is necessary to study new approaches that fit better with new emerging technologies. Specifically, we consider that the data-centric publish/subscribe approach brings benefits that have not been reported in the bibliography.

1.2. Goals

The main goals of this Thesis are divided in two groups:

- Distribution and management of multimedia content

  Goal: Design a system to process and distribute multimedia content in real time based in the data-centric publish/subscribe apprach.

Goal: Ease the development of distributed multimedia systems.

Goal: To implement and evaluate the capabilities of the designed system.

- Distributed resource monitoring

  Goal: Design a real time resource monitoring system based in the data-centric publish/suscribe paradigm.

  Goal: To ease the management and deployment of distributed systems.

  Goal: To implement and evaluate the proposed system by direct comparison with other systems available on bibliography.

1.3. Solution overview

The solution described in this Thesis is organized in two parts:

1. **Distributed multimedia system in real time** We propose the use of a data-centric approach to build distributed multimedia processing systems. Thanks to this system it would be possible to implement systems that process multimedia streams in real time, while satisfying the requirements and quality of services needed by this kind of data. In addition, it will be possible a full integration with legacy technology and devices.

2. **Real time monitoring** We propose a distributed monitoring system in real time based in the data centric publish/subcribe approach. This system will allow to monitor available resources in highly dynamic environments. The proposed systems will allow to access the usage statistics of the resources to multiple collection nodes, each of them with their own interests and requirements.

1.4. Main contributions

The main contributions of this Thesis are:

- The design, implementation and evaluation of EMDS, a system for distribution and processing of multimedia stream in distributed environments.

- The design, implementationa and evaluation of DARGOS, a system for real-time resource monitoring in real time for highly dynamic environments.

We consider that the contributions previously exposed are a important step in the state of the art both in the field of distributed processing of multimedia streams, and distributed monitoring systems.

1.4.1. Works psublished

Part of the work presented in this Thesis is available to the research community in indexed journals and conference proceedings.

Contributions directly related with this Thesis:

1. J. M. Lopez-Vega, J. Sanchez-Monedero, **J. Povedano-Molina**, and J. M. Lopez-Soler, "QoS Policies for Audio/Video Distribution Over DDS Middleware," in *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, 2008.

2. J. M. Lopez-Vega, **J. Povedano-Molina**, J. Sanchez-Monedero, and J. M. Lopez-Soler, "Políticas de QoS en una Plataforma de Trabajo Colaborativo sobre Middleware DDS," in *XIII Jornadas de Tiempo Real*, 2010.

3. **J. Povedano-Molina**, J. M. Lopez-Vega, and J. M. Lopez-Soler, "EMDS: an Extensible Multimedia Distribution Service," in *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, 2010.

4. **J. Povedano-Molina**, J.M. Lopez-Vega, G. Pardo-Castellote, J.M. Lopez-Soler, "Video Streaming with the OMG Data Distribution Service" en *Real-Time Innovations Inc. (Whitepaper)*, 2010.

5. A. Corradi, L. Foschini, **J. Povedano-Molina**, and J. M. Lopez-Soler, "DDS-enabled Cloud management support for fast task offloading," *2012 IEEE Symposium on Computers and Communications (ISCC)*, pp. 000067–000074, Jul. 2012.

6. **J. Povedano-Molina**, J.M. Lopez-Vega, J.M. Lopez-Soler, A. Corradi, L. Foschini, "DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant clouds", *Future Generation Computer Systems* (2013), [1] (Q1, IF: 1.978)

7. Marcello Cinque, Antonio Corradi, Luca Foschini, Flavio Frattini, **Javier Povedano-Molina**, "Scalable Monitoring and Dependable Job Scheduling Support for Multi-domain Grid Infrastructures" *IEEE Communications Magazine: Special Issue Monitoring and Troubleshooting Multi-domain Networks using Measurement Federations* (ENVIADO)

[1] http://dx.doi.org/10.1016/j.future.2013.04.022

Other contributions partially related with this Thesis:

1. **J. Povedano-Molina**, J. M. Lopez-Vega, J. Sanchez-Monedero, and J. M. Lopez-Soler, "Instant Messaging Based Interface for Data Distribution Service," in *XIII Jornadas de Tiempo Real*, 2010.

2. Jose M. Lopez-Vega, **Javier Povedano-Molina**. Gerardo Pardo-Castellote. Juan M. Lopez-Soler. A content-aware bridging service for publish/subscribe environments Journal of Systems and Software 2012. doi:10.1016/j.jss.2012.07.033

3. Juan M. Lopez-Soler, Jose M. Lopez-Vega, **Javier Povedano-Molina**, and Juan J. Ramos-Munoz, . Performance Evaluation of Publish/Subscribe Middleware Technologies for ATM (Air Traffic Management) Systems Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems, 2012.

4. Javier Sanchez-Monedero, **Javier Povedano-Molina**, Jose M. Lopez-Vega, Juan M. Lopez-Soler. Bloom Filter Based Discovery Protocol for DDS Middleware Journal of Parallel and Distributed Computing, Volume 71, Issue 10, October 2011, Pages 1305-1317, ISSN 0743-7315, 10.1016/j.jpdc.2011.05.001. 2011.

5. Javier Sanchez-Monedero, **Javier Povedano-Molina**, Jose M. Lopez-Vega, Juan M. Lopez-Soler. An XML-Based Approach to the Configuration and Deployment of DDS Applications Real-time and Embedded Systems Workshop 2008.

Additionally, part of the work on this Thesis has been also presented in diverse renowned forums because of the interest aroused by some of its results:

1. *Jornadas de Interoperabilidad para Evaluar la Tecnología DDS*, organized by the *Jefatura de Sistemas de la Información, Telecomunicaciones y Asistencia Técnica (JCISAT)* of the Ministry of Defense of Spain where a first prototype of EMDS, one of the systems proposed in this Thesis was presented (Madrid, October 2008).

2. Presentation "Video Over DDS" carried out during the NATO panel meeting "Information Systems Technology de la NATO Research & Operation" (RTO-IST-090), in the *Instituto Tecnológico de la Marañosa* (Madrid, October 2010).

1.5. Document structure

The remainder of the document is organized as follows.

In the Chapter 2 a study of the state of the art in publish/subscribe, multimedia systems and resouce monitoring is carried out.

After that, the solution proposed by this Thesis is presented. For a better understanding it has been divided into two chapters:

In Chapter 3, EMDS, an architecture aimed to build distributed multimedia processing and distribution systems in real time, is presented and assessed.

In Chapter 4, DARGOS a system for real time resource monitoring in highly dynamic systems is presented and evaluated.

Finally, in Chapter 5 the main conclusions of the Thesis and some future work are summarized.

1 Introducción

1.1. Motivación

El mundo que nos rodea puede ser representado mediante flujos multimedia. Desde hace años toda la información que vemos y oímos puede ser capturada y transmitida por dispositivos digitales hasta llegar a localizaciones alejadas geográficamente. Sin embargo, los avances tecnológicos recientes han hecho posible la construcción de sistemas cada vez mejores que permiten procesar dicha información multimedia y ponerla a disposición del usuario en un abanico de dispositivos que van desde ordenadores personales hasta dispositivos móviles.

Uno de los campos que más ha avanzado en este sentido ha sido la construcción de sistemas que procesan dichos flujos multimedia para la extracción de información y que la adaptan a los distintos dispositivos que nos acompañan en nuestra vida diaria. Así por ejemplo, es posible grabar un vídeo y que esté disponible para múltiples usuarios en tiempo real independientemente del dispositivo empleado. Para ello, es necesario disponer de servicios que por ejemplo conviertan dicho flujo a diferentes resoluciones y formatos.

Tradicionalmente, los servicios multimedia se proporcionan en servidores centrales cuya responsabilidad es realizar todo el procesamiento necesario. A pesar de su elevada aceptación, el modelo centralizado –basado en patrones de solicitud/respuesta– presenta una serie de limitaciones, tales como la escalabilidad y la necesidad de conocer la localización de dichos servicios (acoplamiento espacial).

Recientemente han surgido nuevos paradigmas que buscan eliminar el acoplamiento existente entre los contenidos y la localización física del servidor que proporciona dichos contenidos, dando lugar a lo que se conoce como *named-base networking* [90].

En este sentido en el Request For Proposal (RFC) "Architectural Principles of the Internet" [23] (en su Sección 4.1) se comenta literalmente:

– "In general, user applications should use names rather than addresses."

De acuerdo con esta afirmación, los usuarios no deberían acceder a los recursos usando la dirección física del dispositivo que los alberga, sino que deberían poder acceder a los mismos por medio de un nombre que los identifique. En el caso de los contenidos multimedia, las ventajas de esta nueva aproximación son importantes:

Descubrimiento. El descubrimiento e identificación de los contenidos multimedia disponibles sería mucho más sencillo, ya que no estarían asociados a una dirección física. De este modo se permitiría un *acceso basado en contenido*.

Movilidad. Los contenidos podrían ser accedidos independientemente de dónde hubieran sido originados, permitiendo así que los servicios puedan ser duplicados o migrados en tiempo real de una manera transparente y posiblemente sin interrupciones.

Extensibilidad. Al desacoplar el contenido de la localización física (dirección Internet Protocol (IP)), se podría simplificar la provisión de nuevos servicios recursivos, tal que estos utilizaran los contenidos generados por otros servicios, y a su vez estos nuevos podrían ser igualmente consumidos por otros servicios.

Un modelo de interacción que están tomando especial relevancia recientemente es el paradigma de publicación/suscripción. En este paradigma, los consumidores de información (suscriptores) indican qué información desean recibir mientras los productores de información (publicadores) se encargan de dirigir los contenidos a los suscriptores interesados en la información que publican.

Recientes trabajos en el campo del procesamiento y difusión del contenido multimedia, demuestran la aplicabilidad y conveniencia de este nuevo paradigma [22, 43, 97, 106, 183, 202].

Dentro de los sistemas de publicación/suscripción, existe una variante denominada centrada en datos (*data-centric*) que permite aplicar un nuevo enfoque en la distribución de contenidos. En el modelo publicación/suscripción centrada en datos (*data-centric publish/subscribe*) el formato («*typeado*») y contenido de los datos son los que dirigen las operaciones relacionadas con el intercambio de información entre publicadores y suscriptores. En este sentido, la relativamente reciente especificación Data Distribution Service (DDS) normalizada por el Object Management Group (OMG) ha servido para que esta tecnología se convierta en una referencia en este tipo de comunicaciones.

DDS es actualmente el núcleo de las comunicaciones en multitud de sistemas que abarcan desde sistemas financieros hasta entornos tácticos militares [152, 153]. Estos escenarios tienen en común una serie de características, que se detallan a continuación:

- Están compuestos por múltiples productores y consumidores de datos distribuidos geográficamente, alcanzando en ocasiones varios miles de ellos.
- Estos sistemas están continuamente sujetos a cambios y evoluciones, debido a la aparición de nuevos participantes o requisitos.
- La distribución de los datos ocurre en tiempo real, resultando crítico reducir en lo posible los retrasos de transmisión.
- Normalmente, trabajan con grandes volúmenes de datos que fluyen continuamente entre productores y consumidores de información.
- Son escenarios en los que la robustez, fiabilidad y altas prestaciones del sistema de comunicación es un requisito.

Las anteriores características son compartidas por los sistemas de provisión de servicios multimedia. En este tipo de escenarios existen múltiples nodos que consumen flujos provenientes de dispositivos geográficamente dispersos y que a su vez producen nuevos flujos multimedia –típicamente tras alguna transformación o procesado–. Dichos flujos –normalmente implican grandes volúmenes de información– son muy exigentes en cuanto a demanda de recursos y son sensibles a los retardos.

En esta Tesis se estudiarán las ventajas que aporta este nuevo paradigma para la construcción de sistemas multimedia distribuidos robustos, fiables y de altas prestaciones en tiempo real.

Un inconveniente en el procesamiento distribuido de flujos multimedia es que una incorrecta gestión de los recursos puede acarrear –por ejemplo– incrementos de retardo o *jitter*. Este hecho significará una degradación de calidad en el servicio proporcionado. Así por ejemplo, si un sistema distribuido de detección de objetos (*tracking*) de vídeo presentara sobrecarga en alguno de sus nodos, podría causar retardos no tolerables e incluso detecciones falsas. Para paliar esta potencial debilidad, sería deseable poder añadir a demanda nuevos nodos al sistema sin interrumpir el servicio prestado. Para ello, determinar cuándo y cómo se ha de realizar esta gestión es de vital importancia, y es aquí donde se necesita un sistema fiable y eficiente de monitorización de recursos.

El problema de la monitorización de recursos en sistemas distribuidos ha retomado recientemente un impulso creciente debido a la generalización de los sistemas de computación en la nube (*cloud computing*) y aunque hay múltiples sistemas

propuestos, por ejemplo [29, 116, 124], pocas veces se ha afrontado este problema adoptando una aproximación como la que se propone en esta Tesis, consistente en aprovechar los beneficios que la aproximación *data-centric* y el paradigma publicación/subscripción pueden aportar.

1.1.1. Procesamiento multimedia distribuido

El procesamiento multimedia distribuido permite la construcción de sistemas para la transformación de distintos tipos de flujos (audio, vídeo, etc) provenientes de diversas fuentes en otros flujos derivados o la extracción de información a partir del análisis del contenido de los mismos.

A continuación se identifican algunos escenarios en los que el procesamiento de flujos multimedia es de vital importancia y que son potenciales beneficiarios de los resultados y contribuciones de esta Tesis:

Unmanned Aerial Vehicles (UAV). Los vehículos aéreos no tripulados o UAV son aeronaves sin tripulación que están diseñadas para la captura/monitorización de información multimedia en entornos remotos. Normalmente se usan con fines militares tales como misiones de reconocimiento o ataque, pero son cada vez más comunes en otras aplicaciones civiles como la lucha contra incendios, gestión de cultivos o operaciones de búsqueda y rescate (Search and Rescue (SAR)) [66]. La información capturada por las aeronaves se utiliza para la toma de decisiones desde tierra. Típicamente, los UAV disponen de recursos limitados. Por este motivo, es necesario transmitir la información capturada a los diversos puntos de toma de decisiones de una manera eficiente. En estos escenarios es habitual que la información sea visualizada y procesada por distintos nodos a la vez, lo que permiten la toma de decisiones coordinadas. Ejemplos de procesamiento o servicios multimedia en estos entornos son el uso de algoritmos de seguimiento de objetos en movimiento o la detección de anomalías desde el aire y su etiquetado en tierra.

Vídeo Vigilancia. Los sistemas de vídeo vigilancia o Closed Circuit TeleVision (CCTV) consisten en un conjunto de vídeo cámaras desplegadas que permiten la supervisión de un determinado entorno o evento. Originalmente, estas cámaras se encontraban directamente conectadas a monitores, pero la evolución de los dispositivos electrónicos ha permitido la interconexión por medio de red con otros dispositivos como computadoras o dispositivos móviles usados por personal de vigilancia. Entre las funcionalidades que debe de cubrir este sistema se encuentran la visualización en dispositivos con capacidades heterogéneas (con limitaciones en resolución y ancho de banda,etc), almacenamiento de material audiovisual y detección de anomalías o comportamientos sospechosos, así como la notificación de los mismos [21]. Estas funcionalidades necesitan distintos servicios desplegados en el sistema, tales

que puedan descubrir y procesar los distintos flujos de vídeo y datos que están siendo capturados por las cámaras y sensores. Escenarios de aplicación más concretos son el control de fronteras o el análisis de tráfico rodado.

Eventos deportivos. En determinados eventos deportivos, es de interés la integración de la información audiovisual con otros datos provenientes de sensores que debe de ser superpuesta y sincronizada con la imagen. Un ejemplo es una carrera de Fórmula 1. En este escenario existen varios vehículos equipados con distintas cámaras y sensores que capturan datos de telemetría de diversa índole, tales como la velocidad del vehículo, temperaturas, velocidad del viento así como otros parámetros de la configuración del vehículo. Algunos de estos datos deben integrqarse en tiempo real con los distintos flujos de audio y vídeo provenientes de micrófonos y cámaras instaladas. Ello permite mejorar tanto la experiencia de los espectadores como la toma de decisiones de los ingenieros.

Entornos de colaboración. Otro entorno donde es de utilidad el procesamiento de flujos distribuido es la colaboración remota o videoconferencia. En este tipo de escenarios, múltiples usuarios se conectan por medio de micrófonos y cámaras en una sala virtual donde pueden participar todos compartiendo además imágenes estáticas, aplicaciones o pizarras virtuales[68]. Un problema que presentan estos sistemas es que al permitir la participación de múltiples usuarios al mismo tiempo, demandan mucho ancho de banda. Este hecho se agrava cuando se utilizan dispositivos con recursos y capacidades heterogéneas. Una solución a este problema es usar servicios que se encarguen de combinar múltiples flujos de información audiovisual en un único flujo que contenga toda la información de relevancia: por ejemplo mezclando las distintas señales de audio y combinando los vídeos en una cuadrícula. Otra solución pasa por desplegar servicios que creen copias en baja resolución de los flujos multimedia.

El elemento común de todos los escenarios anteriormente expuestos, es que además de productores y consumidores de flujos multimedia, requieren de un despliegue de servicios multimedia encargados de realizar tareas de procesamiento y transformación de los mismos. Normalmente, estos servicios se despliegan utilizando aproximaciones centralizadas que deben de ser diseñadas *ad-hoc* y requieren de importante intervención humana cuando deben de ser modificados o extendidos con nuevas funcionalidades o fuentes de datos multimedia.

Por este motivo, esta Tesis investiga el uso de una aproximación publicación/-suscripción centrada en datos para así solventar las carencias que presentan los sistemas centralizados.

1.1.2. Monitorización de sistemas distribuidos

Uno de los problemas de vital importancia en los sistemas distribuidos es el control del estado y uso de los recursos (tales como el uso de la Central Processing Unit (CPU), memoria, uso de red, temperaturas, accesos a disco, etc). Esto es especialmente necesario en escenarios donde exista una alta carga computacional variable a lo largo del tiempo, como así ocurre en los sistemas de provisión y procesamiento de flujos multimedia.

Los sistemas de monitorización distribuida normalmente constan de un servidor central encargado de recopilar las estadísticas de uso de recursos mediante peticiones explícitas a los nodos de procesamiento existentes. A pesar de ser una aproximación válida para algunos entornos, hay que destacar que además de sufrir los problemas ya mencionados inherentes a los sistemas centralizados, deben paliar las siguientes dificultades:

Heterogeneidad de requerimientos. Existen multitud de aplicaciones interesadas en la información de monitorización, cada una con sus propios requerimientos de actualización. Mientras aplicaciones interactivas como un panel de administración necesitan recibir actualizaciones de uso de estadísticas a una frecuencia lo suficientemente baja como para ser asimilada por operadores humanos, otras aplicaciones más especializadas como puede ser balanceadores de carga, necesitan actualizaciones a una mayor frecuencia para la toma de decisiones.

Variabilidad. Muchos de los sistemas centralizados de monitorización están diseñados para gestionar entornos poco cambiantes. La adición de nuevos nodos de monitorización supone un proceso de configuración en el que o bien se especifica al servidor los nuevos nodos a monitorizar, o bien estos nodos deben conocer la dirección del servidor central donde enviar la información.

La irrupción de nuevos paradigmas de computación como el *Cloud Computing*, en los que grandes infraestructuras (reales y virtuales) se gestionan dinámicamente bajo demanda, ha hecho que los servicios centralizados de monitorización no sean adecuados en algunos casos. Por este motivo, los nuevos paradigmas de computación demandan nuevas aproximaciones que satisfagan las necesidades de estas tecnologías emergentes. Concretamente, esta Tesis también estudia cómo y qué beneficios pueden obtenerse al adoptar una aproximación de publicación/suscripción centrada en datos para la monitorización de recursos en sistemas distribuidos.

1.2. Objetivos

Los objetivos de esta Tesis se dividen en dos grandes grupos:

- Provisión de servicios, Gestión y distribución de contenido multimedia.

  Objetivo. Diseñar un sistema para la provisión de servicios, procesamiento y distribución de contenido multimedia en tiempo real basado en la aproximación publicación/suscripción centrada en datos.

  Objetivo. Facilitar el desarrollo de sistemas multimedia distribuidos.

  Objetivo. Implementar y evaluar las prestaciones del sistema diseñado.

- Monitorización de recursos distribuidos.

  Objetivo. Diseñar un sistema de monitorización de recursos en tiempo real basado en la aproximación publicación/suscripción centrada en datos.

  Objetivo. Facilitar la gestión de sistemas distribuidos.

  Objetivo. Implementar y evaluar el sistema diseñado comparándolo con otros sistemas de referencia.

1.3. Resumen de la solución propuesta

La solución descrita en esta Tesis se organiza en dos grandes apartados:

1. **Sistema multimedia en tiempo real distribuido.** Proponemos el uso de una aproximación publicación/subscripción centrada en datos para la construcción de sistemas de procesamiento distribuido de contenido multimedia. Gracias a la arquitectura propuesta se podrán implementar sistemas robustos, fiables y eficaces que procesen los flujos multimedia en tiempo real, satisfaciendo los requerimientos de este tipo de datos. Además el sistema permitirá su integración total con los dispositivos y tecnologías existentes en la actualidad.
2. **Monitorización de recursos en tiempo real.** Proponemos un sistema de monitorización distribuido en tiempo real basada en la aproximación publicación/suscripción centrada en datos. Este sistema permitirá monitorizar los recursos disponibles en entornos distribuidos cambiantes a lo largo del tiempo. La aproximación empleada gracias a su robustez y escalabilidad permitirá el acceso a las estadísticas de uso de los recursos disponibles en grandes despliegues, con múltiples nodos de recolección, cada uno de ellos con sus propios intereses y requerimientos de monitorización.

1.4. Contribuciones de esta tesis

Las principales contribuciones de esta tesis son:

- El diseño, implementación y evaluación de Extensible Multimedia Distribution Service (EMDS), un sistema para distribución y procesamiento de flujos multimedia en entornos distribuidos.
- El diseño, implementación y evaluación de Distributed Architecture for Resource manaGement and mOnitoring in cloudS (DARGOS), un sistema para la monitorización de recursos en tiempo real para entornos distribuidos altamente dinámicos.

Las contribuciones anteriormente expuestas, tras las fases de diseño y su posterior implementación, han sido evaluadas experimentalmente. Las evaluaciones realizadas permiten concluir que los dos sistemas propuestos implican un avance en el estado del arte tanto en el campo del procesamiento distribuido de flujos multimedia, como en la monitorización de sistemas distribuidos.

1.4.1. Trabajos publicados

Parte de nuestro trabajo se encuentra disponible para la comunidad científica a través de distintas comunicaciones y publicaciones, algunas indexadas en Journal Citation Reports (JCR).

Publicaciones directamente relacionadas con el objeto de esta Tesis:

1. J. M. Lopez-Vega, J. Sanchez-Monedero, **J. Povedano-Molina**, and J. M. Lopez-Soler, "QoS Policies for Audio/Video Distribution Over DDS Middleware," en *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, 2008.

2. J. M. Lopez-Vega, **J. Povedano-Molina**, J. Sanchez-Monedero, and J. M. Lopez-Soler, "Políticas de QoS en una Plataforma de Trabajo Colaborativo sobre Middleware DDS," en *XIII Jornadas de Tiempo Real*, 2010.

3. **J. Povedano-Molina**, J. M. Lopez-Vega, and J. M. Lopez-Soler, "EMDS: an Extensible Multimedia Distribution Service," en *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, 2010.

4. **J. Povedano-Molina**, J.M. Lopez-Vega, G. Pardo-Castellote, J.M. Lopez-Soler, "Video Streaming with the OMG Data Distribution Service" en *Real-Time Innovations Inc. (Whitepaper)*, 2010.

5. A. Corradi, L. Foschini, **J. Povedano-Molina**, and J. M. Lopez-Soler, "DDS-enabled Cloud management support for fast task offloading," *2012 IEEE Symposium on Computers and Communications (ISCC)*, pp. 000067–000074, Jul. 2012.

6. **J. Povedano-Molina,** J.M. Lopez-Vega, J.M. Lopez-Soler, A. Corradi, L. Foschini, "DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant clouds", *Future Generation Computer Systems* (2013), http://dx.doi.org/10.1016/j.future.2013.04.022 (Q1, IF: 1.978)

7. Marcello Cinque, Antonio Corradi, Luca Foschini, Flavio Frattini, **Javier Povedano-Molina**, "Scalable Monitoring and Dependable Job Scheduling Support for Multi-domain Grid Infrastructures" *IEEE Communications Magazine: Special Issue Monitoring and Troubleshooting Multi-domain Networks using Measurement Federations* (ENVIADO)

Otras contribuciones parcialmente relacionadas con esta Tesis:

1. **J. Povedano-Molina**, J. M. Lopez-Vega, J. Sanchez-Monedero, and J. M. Lopez-Soler, "Instant Messaging Based Interface for Data Distribution Service," en *XIII Jornadas de Tiempo Real*, 2010.

2. Jose M. Lopez-Vega, **Javier Povedano-Molina**. Gerardo Pardo-Castellote. Juan M. Lopez-Soler. A content-aware bridging service for publish/subscribe environments Journal of Systems and Software 2012. doi:10.1016/j.jss.2012.07.033

3. Juan M. Lopez-Soler, Jose M. Lopez-Vega, **Javier Povedano-Molina**, and Juan J. Ramos-Munoz, . Performance Evaluation of Publish/Subscribe Middleware Technologies for ATM (Air Traffic Management) Systems Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems, 2012.

4. Javier Sanchez-Monedero, **Javier Povedano-Molina**, Jose M. Lopez-Vega, Juan M. Lopez-Soler. Bloom Filter Based Discovery Protocol for DDS Middleware Journal of Parallel and Distributed Computing, Volume 71, Issue 10, October 2011, Pages 1305-1317, ISSN 0743-7315, 10.1016/j.jpdc.2011.05.001. 2011.

5. Javier Sanchez-Monedero, **Javier Povedano-Molina**, Jose M. Lopez-Vega, Juan M. Lopez-Soler. An XML-Based Approach to the Configuration and Deployment of DDS Applications Real-time and Embedded Systems Workshop 2008.

Adicionalmente, el trabajo realizado en esta Tesis ha sido presentado en diversos foros de reconocido prestigio debido al interés suscitado por los resultados de la misma:

1. Jornadas de Interoperabilidad para Evaluar la Tecnología DDS, organizadas por la Jefatura de Sistemas de la Información, Telecomunicaciones y Asistencia Técnica (JCISAT) del Ministerio de Defensa de España y donde se presentó un primer prototipo de EMDS, el sistema propuesto en esta Tesis (Madrid, Octubre 2008).

2. Presentación "Video Over DDS" realizada en las Jornadas del grupo "Information Systems Technology de la NATO Research & Operation" (RTO-IST-090), organizadas en el Instituto Tecnológico de la Marañosa (Madrid, Octubre de 2010).

1.5. Estructura del documento

A continuación se describe la organización del presente documento.

En el Capítulo 2 se realiza un recorrido por el estado del arte en publicación/-suscripción, sistemas multimedia y monitorización de recursos, tres campos íntimamente relacionados con esta Tesis.

A continuación se describirá la solución propuesta en esta Tesis. Para una mejor comprensión de la misma, el contenido se ha dividido en dos capítulos.

En el Capítulo 3 se presenta EMDS, una arquitectura destinada a la construcción de sistemas de difusión y procesamiento de contenido multimedia en tiempo real.

En el Capítulo 4 se presenta DARGOS, un sistema de monitorización de recursos en tiempo real para sistemas distribuidos altamente dinámicos.

Para finalizar, en el Capítulo 5 se resumen las conclusiones del trabajo realizado y posibles líneas de trabajo futuro.

2

Trabajo Relacionado

Teniendo en cuenta los objetivos de esta Tesis, este Capítulo incluye una revisión de los sistemas, tecnologías y propuestas más relevantes relacionados directamente con la investigación realizada. En particular, la información proporcionada se ha estructurado de la siguiente manera. En la primera sección se incluye una revisión de los sistemas de publicación/subscripción más difundidos. En la siguiente sección se revisan los aspectos y sistemas de distribución multimedia más relevantes. A continuación se plantea el problema de la monitorización de los sistemas distribuidos en general y más concretamente en *Cloud*. En definitiva, se establece el estado del arte de las tecnologías y aportaciones hasta ahora más significativas relacionadas con los objetivos de esta Tesis.

2.1. Publicación/subscripción

Tradicionalmente, las aplicaciones distribuidas en entornos de red adoptan una aproximación cliente/servidor. Típicamente, el cliente inicia la comunicación haciendo peticiones a un servidor, el cual devuelve una respuesta acorde a dicha petición. Un ejemplo -entre otros muchos- de este tipo de interacción es un cliente web, el cual para acceder a la información deseada deberá solicitarla al servidor correspondiente, que la enviará como respuesta.

Este esquema de interacción es adecuado cuando la cardinalidad de las partes interesadas es uno a uno (1:1) y siempre que la interacción sea de naturaleza síncrona. Sin embargo, esto no es siempre así, ya que existen múltiples escenarios en los que varios clientes pueden estar interesados en la misma información y en los que las entidades no tienen por qué estar necesariamente acopladas (tanto temporal como espacialmente).

Figura 2.1: Paradigma de publicación/subscripción.

Un claro ejemplo –que pone de manifiesto las deficiencias del modelo cliente/servidor– es una aplicación de monitorización del precio de las acciones en el mercado de valores. En un esquema basado en cliente/servidor, cada cliente deberá realizar periódicamente peticiones de actualización a un servidor, el cual devolverá la cotización de dichas acciones. En este caso, por razones obvias, es de vital importancia tener el precio actualizado con el mínimo retardo posible. Esto significa que cada cliente debe sondear a una tasa muy alta al servidor.

A partir de las anteriores consideraciones, en el modelo cliente/servidor –a pesar de su adopción masiva– se pueden identificar los siguientes problemas:

Acoplamiento. El modelo impone un fuerte acoplamiento –tanto espacial como temporal– entre las entidades. El acoplamiento espacial implica que el cliente debe conocer la dirección IP y puerto del servidor en el que se aloja el recurso o servicio de interés. En el dominio temporal, el acoplamiento exige que el productor de la información (servidor) debe coincidir en el tiempo con el consumidor (cliente), de otro modo no se podrá realizar la comunicación.

Sincronismo. El modelo cliente/servidor normalmente va ligado al sistema de interacción petición/respuesta. Sin embargo, algunos patrones de interacción asíncronos –tales como la notificación de eventos en tiempo real– no son eficientes si para ellos se adopta un modelo petición/respuesta. Esto es debido a que en este modelo, el cliente debe sondear continuamente al servidor acerca de la ocurrencia de un determinado evento, generando –por tanto– gran cantidad de tráfico de red. Esto puede significar que la aplicación cliente tenga que sacrificar precisión temporal en la notificación, para evitar así la posible saturación del servidor.

Robustez. El modelo cliente/servidor es inherentemente centralizado, si el servidor/productor de información presenta algún fallo, la estabilidad de todo el sistema se puede ver comprometida.

Para los escenarios en los que el modelo cliente servidor no es necesariamente adecuado y para mitigar las dificultades antes identificadas, surgen otros modelos como el de publicación/subscripción [56]. En este modelo son los productores de información (denominados publicadores) los que notifican las actualizaciones a los consumidores de información (también denominados como subscriptores).

En este modelo, los suscriptores deben expresar previamente qué información desean recibir, siendo responsabilidad del servicio de publicación/subscripción hacer llegar a cada subscriptor únicamente los datos publicados que satisfagan los intereses de dicho subscriptor (Figura 2.1).

Según [56], los sistemas de publicación/subscripción se pueden clasificar considerando el criterio empleado por los subscriptores al seleccionar los eventos/datos (publicaciones) a recibir. En concreto, los sistemas de publicación/subscripción se pueden clasificar atendiendo a los siguientes criterios:

Basados en tópicos (*topic-based*): En este tipo de sistemas las publicaciones se identifican mediante un nombre único o tópico. De acuerdo con esto, los subscriptores deben indicar el nombre de las publicaciones que quieren recibir. Para proporcionar mayor flexibilidad, algunos sistemas usan jerarquías de nombres conjuntamente con expresiones regulares [83, 176].

Basados en contenido (*content-based*): En los sistemas basados en contenido, los subscriptores indican su interés en recibir aquellas publicaciones que cumplan cierto criterio. Para ello, los subscriptores indican mediante un lenguaje de subscripción específico y operadores de comparación, el criterio que deben cumplir los datos a recibir [72, 92].

Basados en tipo (*type-based*): En este caso, los eventos publicados tienen una estructura determinada (tipo de datos). En este caso, los subscriptores indican su interés en publicaciones que tengan un determinado tipo de datos[55, 139].

En los siguiente apartados se identifican y explican brevemente los esquemas de publicación/subscripción más relevantes, se destacan sus características principales, así como sus ventajas y limitaciones. Concretamente, se estudia *Message Queue Telemetry Transport* (MQTT), Java Message Service (JMS), Advanced Message Queuing Protocol (AMQP), DDS, además de otros, finalizando con una comparativa entre los mismos.

2.1.1. *Message Queue Telemetry Transport* (MQTT)

MQTT es un protocolo sencillo de publicación/subscripción basado en tópicos

[83]. Fue diseñado en 1999 por *IBM* en colaboración con *Arcom* (ahora denominada Eurotech).

Este protocolo se diseñó para dispositivos y redes con capacidades reducidas, típicamente con escaso ancho de banda y/o con altas tasas de fallos. En este sentido, MQTT es adecuado para escenarios del tipo *Internet of Things* o *Machine to Machine* (M2M).

MQTT hace uso de servidores intermediarios (*brokers* en adelante) para realizar el intercambio de información entre publicadores y subscriptores. En esta aproximación, tanto publicadores como suscriptores envían información al *broker* acerca de la información que –respectivamente– producen o están interesados en recibir y es dicho *broker* el encargado de que las publicaciones lleguen correctamente a los subscriptores adecuados. Con esta aproximación se consigue que la carga computacional requerida en los dispositivos productores o consumidores de información sea baja, por lo que también está indicado para redes de sensores [81].

Aunque todavía MQTT no tiene formalizado el *estatus* de estándar, se están realizando esfuerzos en ese sentido, ya que ha sido propuesto recientemente como estándar desde el consorcio OASIS [130].

Uno de los hechos más destacables de MQTT es que la especificación es pública y libre de *royalties*, estando actualmente vigente la versión 3.1 [83, 122].

La especificación actualmente disponible abarca tres aspectos del protocolo: el formato de los mensajes MQTT, los tipos de mensajes definidos en el protocolo, así como su utilidad, y el flujo de mensajes entre los publicadores y subscriptores con el *broker*.

Las características más destacables de MQTT son:

- MQTT define un protocolo de comunicación que facilita la interoperabilidad entre distintos implementadores.
- Soporta *multicast*, facilitando así las interacciones en escenarios donde existen múltiples publicadores y subscriptores interactuando.
- Define varias Quality of Service (QoS) básicas, de acuerdo con los requisitos de entrega de una determinada publicación.
- El protocolo MQTT es totalmente agnóstico respecto al tipo de datos que transportan las publicaciones, ya que no se define dentro del estándar el formato de serialización de los mismos.
- El protocolo ha sido diseñado para racionalizar el consumo de ancho de banda. Así por ejemplo, las cabeceras fijas de mensaje constan solo de 2 bytes.

Respecto al uso de políticas QoS en la difusión de publicaciones, MQTT ofrece 3 modos o alternativas diferentes:

At most once Este caso corresponde a una publicación sin confirmación, por tanto no se garantiza que el subscriptor reciba la información libre de errores. Esta alternativa se adopta en publicaciones en las que la pérdida de un mensaje sea relativamente tolerable.

At least once En este caso, la correcta recepción de las publicaciones estará garantizada, aunque no se evita la posible recepción de mensajes duplicados.

Exactly once Este caso es similar a las publicaciones *at least once*, pero con la salvedad de que no se admiten mensajes duplicados.

Uno de los mayores inconvenientes de MQTT es que puede usar únicamente Transmission Control Protocol (TCP) (protocolo de transporte orientado a conexión). Este hecho hace que por diseño, MQTT no soporte conexiones *multicast*, limitando así su escalabilidad y la capacidad de descubrir automáticamente otras entidades MQTT.

2.1.2. Java Message Service (JMS)

Java Message Service (JMS) [73] es un estándar desarrollado por *Sun Microsystems* en el que se especifica una Application Programming Interface (API) para Java destinada al uso de paso de mensajes. En esta API se definen dos modelos principales de interacción: punto a punto y publicación/subscripción. Por su relación con la Tesis, la atención se centra en el segundo de los modelos.

JMS es una API surgida en el año 2001 para facilitar el desarrollo de aplicaciones asíncronas y desacopladas en Java. Este *middleware* se suele considerar dentro de los llamados Message Oriented Middleware (MOM) [10].

Las características básicas de calidad de servicio (QoS) que se soportan en una aplicación JMS son:

Fiabilidad: JMS puede proporcionar transporte de mensajes fiable con retransmisión de mensajes en caso de pérdida.

Transacciones atómicas: Un grupo de mensajes se pude gestionar atómicamente, es decir si uno de ellos falla el sistema puede volver a un estado anterior.

Prioridades: Los mensajes de JMS se pueden priorizar, de modo que el *middleware* puede gestionar primero aquellos considerados como urgentes.

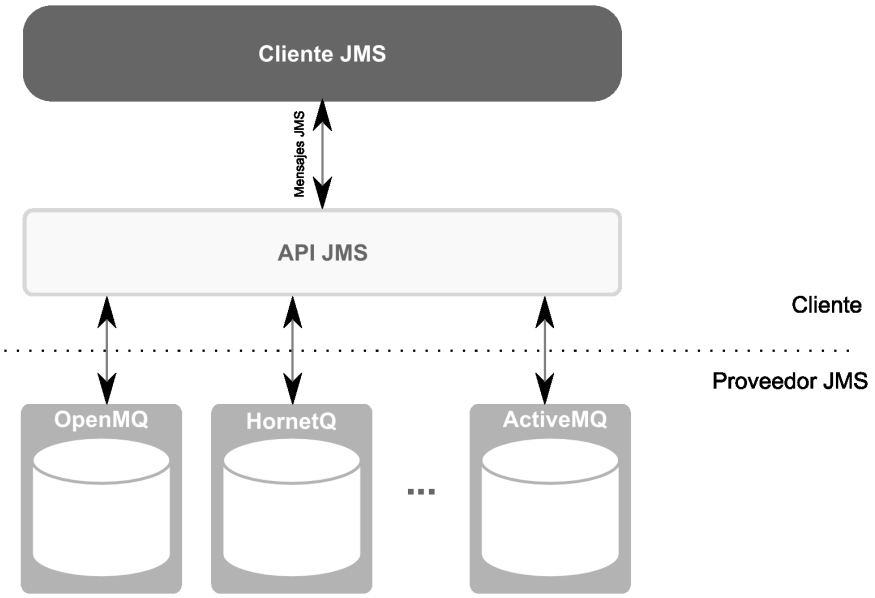


Figura 2.2: Arquitectura JMS.

Duración: Cada mensaje en JMS puede tener asociada una caducidad, a partir de la cual el mensaje no se considera válido.

Uno de los hechos que condiciona el uso de JMS es que la gran mayoría de las implementaciones actuales [8, 91, 137] requieren la instalación de un *broker* encargado de distribuir los mensajes a sus destinatarios. Este hecho implica las siguientes consecuencias:

- Si el volumen de mensajes es elevado, el *broker* se puede convertir en un cuello de botella.
- La latencia extremo a extremo aumenta debido a la intermediación del *broker*.
- Si el *broker* falla, el sistema completo puede verse comprometido.

JMS únicamente especifica la API a emplear a nivel de aplicación (Figura 2.2), pero no estandariza el protocolo de comunicación empleado. Esto es una limitación debido a que no existe interoperabilidad a este nivel entre las distintas implementaciones. Este hecho unido al requerimiento de un *broker* conlleva un inconveniente importante: tanto los clientes (publicadores y subscriptores) como el *broker* deben usar la misma implementación de JMS.

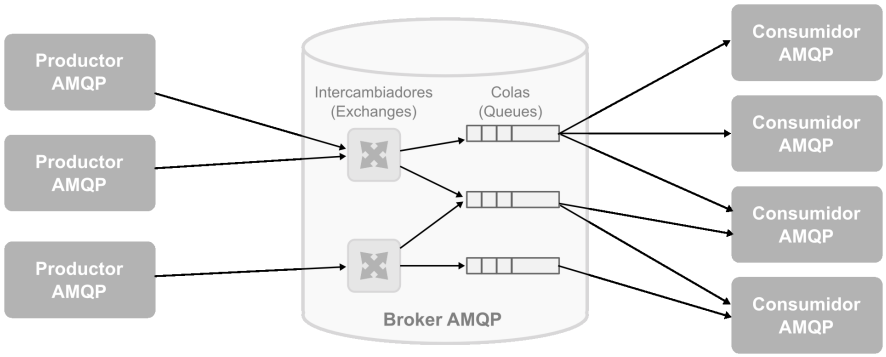


Figura 2.3: Arquitectura AMQP.

2.1.3. Advanced Message Queuing Protocol (AMQP)

El protocolo AMQP [184] es un protocolo estándar abierto orientado a mensajes (MOM) que –al igual que JMS– soporta modelos de comunicación punto a punto y publicación/subscripción. A diferencia del anterior, AMQP estandariza el protocolo de transporte (*wire-protocol*) a utilizar, lo que definitivamente mejora la interoperabilidad: cualquier programa capaz generar y consumir información siguiendo dicho protocolo puede interoperar con otras aplicaciones, con independencia de la implementación.

La especificación de AMQP define: un sistema de tipos, un protocolo de transferencia de mensajes entre procesos, un formato de mensaje estándar y un conjunto de capacidades de distribución de mensajes. Para ello, la arquitectura de AMQP (Figura 2.3) se basa en los siguientes componentes:

***Producer*:** Es una aplicación cliente que genera mensajes AMQP.

***Consumer*:** Es una aplicación cliente que recibe mensajes AMQP generados por un *producer*.

***Broker*:** Es un servidor al que las aplicaciones cliente (*Producers* y *Consumers*) se conectan para enviar y recibir mensajes.

***Queue*:** Representa una abstracción de una cola, que se utiliza para almacenar y enviar los mensajes a los consumidores adecuados.

***Exchange*:** Este componente es encargado de redirigir y filtrar los mensajes enviados por los *Producers*.

El modelo basado en colas (*Queues*) e intercambiadores (*Exchanges*) de AMQP permite gestionar de forma individualizada cada mensaje publicado. Por ejemplo, se

pueden configurar parámetros como la caducidad o la persistencia de los mensajes, en caso de fallo en el servidor. Esto es posible gracias al nivel extra de indirección que introduce la presencia de intercambiadores (*Exchanges*): en lugar de establecer directamente una conexión entre *Producers* y *Consumers*, los *Producers* envían un mensaje al *Exchange*, que a su vez estará encargado de reenviarlo –en función de los parámetros– a una o múltiples colas a las que se asocian los *Consumers*.

Esta arquitectura exime a los publicadores de conocer explícitamente: la dirección de cada uno de los subscriptores, la cola (*Queue*) asociada a cada uno de ellos y los requisitos específicos de entrega de cada uno de los subscriptores.

En AMQP es posible declarar diferentes tipos de intercambiadores. Además, cada uno tendrá definida una manera particular de interaccionar con las colas del servidor. Estos son los tipos de intercambiadores que propone el estándar AMQP:

Multidifusión (*Fanout Exchange*). Los mensajes que llegan a este intercambiador son redirigidos automáticamente a todas las colas que tenga asociadas dicho intercambiador.

Directo (*Direct Exchange*). Los mensajes enviados a este intercambiador tienen asociada una clave que sirve para redirigirlos automáticamente a la cola que coincida en la clave asociada.

Tópico (*Topic Exchange*). Este es un caso especial de los intercambiadores directos. En este caso, en lugar de una clave concreta, las colas tienen asociadas un patrón.

Cabecera (*Header Exchange*). En este caso en lugar de utilizar un emparejamiento mediante clave para redirigir los mensajes a las colas, se utilizan valores que se encuentran en la cabecera del mensaje. Este mecanismo permite una redirección de mensajes más flexible que el directo, ya que permite redirigirlos en base a criterios que no sean de tipo cadena de texto (*e.g.* números enteros o diccionarios).

Personalizado (*Custom Exchange*). El estándar AMQP deja abierta la posibilidad de desarrollar nuevos intercambiadores adaptados a las necesidades específicas del usuario (*e.g.* un intercambiador que realice reparto de cargas –o *load balancing*–).

La implementación de un servicio de publicación/subscripción se suele hacer típicamente utilizando intercambiadores de tipo *Fanout*. Con este tipo de intercambiadores, cada mensaje publicado es reenviado y recibido por todos y cada uno de los subscriptores. No obstante, mediante el uso adecuado de otros tipos de intercambiadores, se pueden igualmente implementar patrones de publicación/subscripción más complejos, como por ejemplo aquellos que permitan el filtrado de publicaciones.

Respecto a las características de comunicación del estándar AMQP, estás son las más destacables:

Acuse de recibo: El estándar *AMQP* proporciona mecanismos para que a nivel de aplicación se pueda asegurar la correcta recepción y procesamiento de un mensaje (*acknowledgement*).

Durabilidad y persistencia: Indica si una cola o un mensaje debe sobrevivir a un reinicio del *broker*.

Prioridad: Un productor puede asignar una prioridad a cada mensaje, de modo que los mensajes de mayor prioridad son enviados antes que otros mensajes encolados de menor prioridad.

Tiempo de espiración: Cada mensaje puede tener asignado un tiempo máximo de validez (*time to live*), después del cual no debe permanecer en el sistema.

Modelos de comunicación: Los mensajes pueden ser repartidos a los consumidores automáticamente (modelo *push*) o bajo petición explícita (modelo *pull*).

Respecto al transporte de datos empleado, el estándar AMQP considera que todas las comunicaciones se realizan mediante TCP. Este hecho limita su escalabilidad, debido a que el *broker* debe mantener una conexión activa por cada publicador o subscriptor presente en el sistema.

2.1.4. Data Distribution Service (DDS)

Otro de los estándares de publicación/subscripción que ha tomado especial relevancia durante los últimos años es DDS [132, 139].

DDS es una especificación de la OMG que surge de la necesidad de la estandarización de arquitecturas publicación/subscripción en sistemas distribuidos.

Aunque inicialmente estaba orientado a ser un estándar para entornos de tiempo real críticos, tales como entornos militares o financieros [53, 54, 96], DDS ha ido evolucionando hasta ser utilizado en otros escenarios menos críticos aunque exigentes como *Cloud Computing* [6].

A diferencia de otros MOM, una característica esencial de DDS es la adopción de una aproximación centrada en datos (también denominada *data-centric*). Es decir, el contenido de los mensajes transportados no es opaco al *middleware*, sino que este puede inspeccionar el contenido de los mismos y operar en consecuencia.

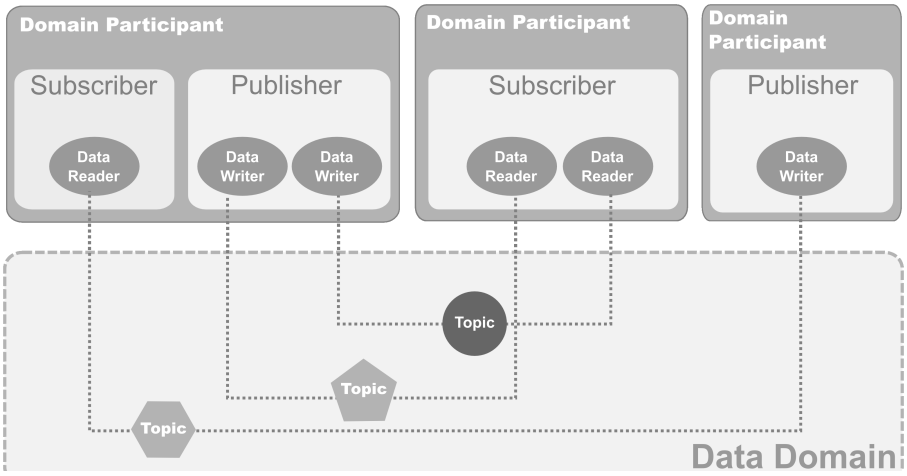


Figura 2.4: Entidades DDS y sus interrelaciones.

La aproximación *data-centric* permite –entre otras ventajas– que el *middleware* pueda filtrar y el almacenar en memoria (cache) los datos. Esto facilita por ejemplo que los subscriptores puedan especificar la recepción de únicamente aquellas muestras cuyos campos cumplan ciertas condiciones (*e.g.* estén dentro de un rango). Esto puede reducir el consumo de ancho de banda y recursos de memoria necesarios.

El carácter centrado en datos del *middleware* tiene como ventaja adicional el posible desacoplo entre aplicaciones diferentes, ya que para comunicarse entre sí, estas solo necesitarán compartir un espacio de datos común, con independencia de la lógica de la propia aplicación.

A diferencia de otras soluciones de publicación/subscripción, la recomendación DDS estandariza tanto las APIs como el protocolo de comunicación empleados. Para ello la OMG define tres capas en la especificación: Data Local Reconstruction Layer (DLRL), Data Centric Publish Subscribe (DCPS) y el Real-Time Publish Subscribe (RTPS). A continuación se hará una breve descripción de cada una de ellas:

2.1.4.1. Data Centric Publish Subscribe (DCPS)

La capa DCPS especifica el modelo de datos, las entidades que participan en una aplicación basada en DDS, sus relaciones y las calidades de servicio que deben ser soportadas. Entidades y conceptos relevantes (Figura 2.4) son:

Dominio (*Domain*). También denominado *Data-Space*, representa a la agregación de datos de distintos tipos provenientes de diversas fuentes. Un dominio aísla las comunicaciones entre aplicaciones distribuidas que necesiten comunicarse. También

se puede considerar como una caché distribuida a la que pueden acceder productores y consumidores de información. Por lo tanto, dos entidades que intercambien información deben pertenecer al mismo *Domain*.

Participante (*Domain Participant*). Representa la intención de una aplicación de intercambiar información dentro de un dominio determinado. Un *Participant* también contiene a las entidades que desean publicar o recibir datos de dicho dominio.

Tópico (*Topic*). Un tópico es la unidad mínima de información que se puede intercambiar de acuerdo a la recomendación *DDS*. Un tópico se define por medio de un nombre único en un dominio y el tipo de datos de la información que contiene. Opcionalmente, distintas instancias de un mismo tópico puede identificarse mediante una clave.

Publicador (*Publisher*). Es la entidad encargada de la difusión de todos los tópicos publicados por una aplicación en un dominio.

Subscriptor (*Subscriber*). Es la entidad encargada de recibir todos los tópicos de interés publicados en un dominio, suministrándolos a la aplicación.

DataWriter. Un *DataWriter* es la entidad encargada de publicar los datos de un tópico determinado dentro de un publicador. Es decir, en cada *Publisher* existe un *DataWriter* distinto para cada tópico que esté siendo publicado.

DataReader. Es el homólogo al *DataWriter* en el lado del *Subscriber*. En este caso, es la entidad encargada de recibir las muestras publicadas de un determinado tópico en un determinado dominio.

Además de las entidades y conceptos listados anteriormente, la capa DCPS especifica las calidades de servicio *QoS* [38] que deben estar disponibles en una implementación que cumpla el estándar *DDS*. A diferencia de otros estándares como AMQP o JMS –que únicamente definen políticas de calidad de servicio simples como la persistencia o fiabilidad– en DDS se definen políticas de calidad de servicio más complejas que permiten optimizar y controlar el uso de recursos locales como la memoria o el ancho de banda consumido. En la Tabla 2.1 se identifican las políticas de calidad de servicio definidas en la especificación DDS. Para cada QoS se detellan 3 caracteristicas: las entidades a las que aplica, si debe de ser negociado entre publicadores y suscriptores, y si puede ser cambiado durante el ciclo de vida de la entidad.

2.1.4.2. Data Local Reconstruction Layer (DLRL)

Esta capa especifica un modelo de datos para realizar las transformaciones necesarias entre tipos de datos nativos de un lenguaje de aplicación dado y el modelo de

QoS Policy	Aplicabilidad	RxO	Modificable	Tipo
DURABILITY	T,DR,DW	Y	N	Data Availability
DURABILITY SERVICE	T,DW	N	N	
LIFESPAN	T,DW	N/A	Y	
HISTORY	T,DR,DW	N	N	
PRESENTATION	P,S	Y	N	Data Delivery
RELIABILITY	T,DW,DR	Y	N	
PARTITION	P,S	N	Y	
DESTINATION ORDER	T,DR,DW	Y	N	
OWNERSHIP	T,DR,DW	Y	N	
OWNERSHIP STRENGTH	DW	N/A	Y	
DEADLINE	T,DR,DW	Y	Y	Data Liveliness
LATENCY BUDGET	T,DR,DW	Y	Y	
TRANSPORT PRIORITY	T,DW	N/A	Y	
TIME BASED FILTER	DR	N/A	Y	Resources
RESOURCE LIMITS	T,DR,DW	N	N	
USER DATA	DP,DR,DW	N	Y	Configuration
TOPIC DATA	T	N	Y	
GROUP DATA	P,S	N	Y	
LIVELINESS	T,DR,DW	Y	N	
WRITER DATA LIFECYCLE	DW	N/A	Y	Lifecycle
READER DATA LIFECYCLE	DR,	N/A	Y	

Tabla 2.1: Políticas de calidad de servicio definidas por el estándar Data Distribution Service (DDS).

datos basado en tópicos proporcionado por el DCPS. Esta capa facilita el desarrollo de aplicaciones ya que permite integrar DDS con los constructores y estructuras típicas de cada lenguaje de programación. No obstante, la implementación de esta capa es opcional, por los que muchos fabricantes no la incluyen en sus distribuciones.

2.1.4.3. Real-Time Publish Subscribe (RTPS)

Uno de los aspectos más importantes para el éxito de un estándar es el de la interoperabilidad. En este aspecto, DDS –al igual que AMQP– especifica el protocolo de red que debe ser utilizado para que así múltiples implementaciones puedan interoperar. Con este fin, dentro de DDS se incluye otro estándar independiente llamado Real-Time Publish Subscribe (RTPS)[133]. De este modo, cualquier aplicación o dispositivo que implemente RTPS –aunque no esté basado en DDS– podrá interoperar con otras aplicaciones o dispositivos que lo soporten. Concretamente, estos son los aspectos y funcionalidades que se especifican en RTPS:

Descubrimiento. El estándar RTPS define un protocolo que permite a una aplicación DDS descubrir automáticamente las entidades presentes en la un espacio de datos determinado. Dicho protocolo permite además descubrir las publicaciones que existen así como sus requerimientos de calidad de servicio.

Formato de mensaje. Las cabeceras y formatos de mensaje son definidos en el estándar RTPS, de modo que cualquier aplicación capaz de formatear e interpretar dichos mensajes pueda interactuar con otras aplicaciones que cumplan el estándar.

Interacción El estándar RTPS determina el secuenciación e interacción de los mensajes RTPS intercambiados entre aplicaciones DDS. De este modo, define mensajes que permiten tanto comunicaciones de tipo *best effort*, como con garantía de recepción (*reliable*).

Formato de datos. Mientras que en otros estándares de publicación/subscripción el formato de codificación de un mensaje es independiente del *middleware* [184], en el caso de RTPS se especifica que el formato por defecto para codificar los mensajes es el estándar Common Data Representation (CDR) [131]. No obstante, la especificación también define mecanismos para permitir al desarrollador utilizar otros formatos de codificación.

2.1.4.4. Características relevantes de DDS

Para concluir y a modo de resumen, de acuerdo con todo lo expuesto anteriormente, estas son las características más destacables de DDS:

El enfoque *data-centric*. Este enfoque es el principal hecho diferenciador respecto de otros estándares de publicación/subscripción. Mientras que en otros MOM, normalmente los datos transportados son opacos al *middleware*, en el caso de DDS el contenido de los mensajes condiciona el flujo de los mismos entre publicadores

y subscriptores. Este hecho permite implementar operaciones de publicación avanzadas, tales como el soporte de filtrado por contenido y la identificación unívoca de muestras de tópicos de acuerdo a una clave basada en el contenido. Gracias a estos mecanismos, se obtienen beneficios, tales como la reducción del tráfico cuando algún subscriptor solo está interesado en muestras cuyo contenido cumpla alguna condición, o mantener una caché de los valores de un tópico publicados recientemente.

Arquitectura descentralizada. El estándar DDS no define explícitamente la arquitectura de su implementación. No obstante, en la especificación de DDS no se menciona la necesidad de un *broker* como así ocurre en JMS, ni tampoco se especifican los requerimientos o estructura que debe satisfacer el *broker* –a diferencia de lo que si ocurre en AMQP–. La ausencia de *brokers* favorece el despliegue de arquitecturas totalmente descentralizadas con componentes totalmente autónomos.

Descubrimiento automático. Como se ha mencionado anteriormente, el estándar RTPS especifica un protocolo de descubrimiento automático. Este protocolo permite a las aplicaciones conocer las publicaciones disponibles y poder subscribirse a ellas. Gracias a este protocolo de descubrimiento no es necesario el despliegue de servicios de directorio donde listar las publicaciones disponibles.

Interoperabilidad. A diferencia de lo que ocurre en otros estándares de publicación/subscripción, la especificación DDS define múltiples capas, que van desde la API, hasta el protocolo de comunicación entre nodos. Este hecho favorece la interoperabilidad con otras aplicaciones, ya que tanto el modelo de datos como el protocolo de comunicación es común para todos los fabricantes.

2.1.5. Otros sistemas de publicación/subscripción

Además de los estándares de publicación/subscripción descritos en la Sección anterior, existen otros sistemas de publicación/subscripción que merecen ser descritos también, pero que su por carácter especializado no tienen cabida en la anterior sección. En este apartado se revisan algunos de los más relevantes.

Publicación/Subscripción en World Wide Web (WWW) La WWW ha estado ligada desde sus comienzos al paradigma cliente/servidor. Típicamente, los clientes acceden a servicios web instalados en un servidor bien localizado.

El incremento y renovación de los contenidos disponibles en la web –sometida a un proceso de actualización permanente– hace que sea ineficiente el sondeo continuo de posibles actualizaciones en servidores. En este sentido, han surgido algunos estándar orientados a la notificación de eventos de manera asíncrona en el entorno de la WWW.

Así por ejemplo, recientemente ha surgido *PubSubHubbub*[58], un especificación de publicación/subscripción para contenidos en la web, mediante la cual los clientes son notificados de la publicación de nuevos contenidos. *PubSubHubbub* permite otorgar capacidades de publicación/ subscripción a los formatos de sindicación de datos *Atom* y Really Simple Sindication (RSS) ampliamente extendidos en sitios de noticias y *blogs*. *PubSubHubbub* define un nuevo componente llamado *hub*, que es el encargado de notificar a los subscriptores cuando un publicador dispone de actualizaciones en su *feed RSS* (*e.g.* una noticia) de modo que el subscriptor realiza entonces la petición de la información actualizada al publicador.

Gracias a este sistema, un agregador de noticias puede ser actualizado casi instantáneamente cuando aparece una noticia nueva, con la particularidad de que no es necesario sondear continuamente al servidor en busca de actualizaciones de contenidos.

Relacionado con esto, ha aparecido recientemente la especificación XEP-0060 [118]. Dicha especificación es una extensión del protocolo eXtensible Mesasaging and Presence Protocol (XMPP) [160] –protocolo descentralizado de amplia difusión basado en eXtensible Markup Language (XML)– que se ha empleado en multitud de escenarios, tales como mensajería instantánea [161], Voice over IP (VoIP), transmisión de vídeo, juegos e Internet de las cosas.

Adicionalmente, por su relevancia también caben destacar los estándares Web Services para la notificación asíncrona de eventos *WS-Notification* y *WS-Topics*. En esta línea, [108] propone un sistema de publicación/subscripción sobre dichas especificaciones.

Finalmente, la tendencia hacia una nueva Internet basada en publicación/suscripción se manifiesta con la aparición de algunos proyectos reciente de aparición como son los proyectos Publish Subscribe Internet Routing Paradigm (PSIRP) y PURSUIT [149, 150].

Publicación/Subscripción en Wireless Sensor Networks (WSN). Otro campo donde recientemente ha habido avances significativos es en el de las Wireless Sensor Networks (WSN). Estas consisten en una infraestructura de red formada por multitud de sensores –de potencia limitada– que se encargan de recopilar mediciones y enviarlas a determinados puntos de recolección –o sumideros– mediante comunicaciones inalámbricas.

Las WSNs se despliegan en diferentes escenarios tales como edificios inteligentes o entornos abiertos para la monitorización ambiental. Este tipo de escenarios se caracterizan por la necesidad de intercambiar de forma asíncrona múltiples mensajes, por lo que pueden ser beneficiarios del paradigma de publicación/subscripción. Este hecho ha originado multitud de trabajos en la literatura relacionados con el uso de publicación/subscripción en entornos WSN.

Por ejemplo, en [26] los autores hacen un estudio del impacto en el uso de *brokers* para dar soporte de publicación/subscripción en WSN. En [193] se propone PS4WSN-a, un *middleware* ligero de publicación/subscripción basado en *broker*. En [105] los autores describen *PSWare*, un *middleware* de publicación/subscripción con soporte de eventos compuestos mediante filtrado inteligente de información. *WMOS* [199] es un *middleware* de publicación/subscripción completamente distribuido que se adapta a la calidad de servicio disponible para hacer un uso eficiente de los recursos. Para ello propone la alternancia entre una estrategia *content-based* y *topic-based* dependiendo de los recursos disponibles. Finalmente, los autores de [168] proponen TinyMQ, un *middleware* basado en la aproximación *content-based*. Este sistema totalmente distribuido se basa en la construcción de una red *overlay* –o recubrimiento– para la distribución de datos basada en el contenido de los mismos.

Señalar que además de las propuestas anteriormente citadas, para intercambiar información mediante el paradigma de publicación/subscripción dentro de WSNs, existen otras basadas en la adaptación de estándares existentes de publicación/subscripción a los requerimientos de los entornos WSN [12, 81], mientras que otras como [15] integran las WSNs con sistemas de publicación/subscripción convencionales.

Publicación/subscripción en Peer to peer (P2P). El auge de las redes y tecnologías P2P también desencadenó la aparición de sistemas de publicación/subscripción especialmente diseñados para ellas. Tal es el caso de *Meghdoot* [72], un sistema de publicación/subscripción basada en contenido. Los autores de [35] propusieron un sistema de publicación/subscripción P2P con soporte integral de calidades de servicio basado en DDS. Finalmente, los autores de [2] definen un protocolo de publicación/subscripción genérico que opera sobre cualquier tecnología P2P, ya sean redes sin estructura [158] o basadas en tablas Hash distribuidas (Distributed Hash Table (DHT)) [173].

2.1.6. Comparativa de estándares

En esta sección se proporciona una comparación directa entre los estándares de publicación/subscripción disponibles. Para ello, el estudio se centra en diferentes aspectos relevantes, tales como el nivel o grado de estandarización, los protocolos

	MQTT	JMS	AMQP	DDS
Arquitectura	Broker [1]	Broker	Broker	Descentralizado [1]
Tipo	Topic	Topic	Topic	Content/Type
API	N	S	N	S
Protocolo	S	N	S	S
Transporte	TCP	TCP	TCP	UDP [2]
QoS	S (3)	S (4)	S (3)	S (20)
Payload	N/A	N/A	N/A	CDR
Filtrado	No	Content	Content	Content/Time

Tabla 2.2: Comparativa entre especificaciones Publicación/Subscripción.

[1] No especificado en la especificación, por lo tanto se especifica el despliegue típico
[2] El estándar permite considerar otros protocolos de transporte

de transporte empleados y las capacidades de los mismos. Concretamente, en la comparativa se han considerado las siguientes características:

Arquitectura. Identifica el modelo de despliegue que el estándar considera por defecto.

Tipo. Se define el tipo de publicación/subscripción realizada, según los criterios definidos en [56].

API. Se refiere a la definición de una API estándar en la especificación.

Protocolo. Se refiere a si se especifica un protocolo con el objetivo de facilitar la interoperabilidad entre distintos fabricantes.

Transporte. Identifica el protocolo de transporte utilizado.

QoS. Identifica si se definen QoS (y su número).

Payload. Formato de serialización de los datos considerados en el protocolo.

Filtrado. Describe la definición en la especificación de mecanismos de filtrado de información por parte de los subscriptores.

En la Tabla 2.2 se observan los resultados de la comparación realizada. En la misma se observa que tanto MQTT, JMS como AMQP tienen características muy similares tanto en modelo de despliegue, el tipo de publicación/subscripción que realizan y el protocolo de transporte utilizado (*e.g.* TCP).

Todos ellos, aunque desacoplan a publicadores y subscriptores, dependen de un *broker* central sin el cual la comunicación no es posible.

Las características diferenciadoras de DDS con respecto al resto de los *middleware* de publicación/subscripción, junto con su aproximación *data-centric* –que permite un mayor control sobre el flujo de información entre nodos– lo hacen más apropiado para el desarrollo de algunas aplicaciones descentralizadas, especialmente en aquellas donde debido al volumen de datos existente es necesario un mayor control sobre los datos recibidos.

Respecto al soporte de calidades de servicio (QoS) cada estándar especifica un conjunto distinto. Así por ejemplo, mientras JMS únicamente especifica un conjunto básico de calidades de servicio como prioridad, confiabilidad y duración, el estándar DDS especifica hasta 20 políticas distintas orientadas a configurar multitud de aspectos del *middleware* (ver la Tabla 2.1). En concreto, publicadores y subscriptores pueden negociar un tiempo mínimo entre actualizaciones de tópicos (*TIME_BASED_FILTER*), garantizar un máximo periodo de tiempo entre actualizaciones (*DEADLINE*), establecer el tiempo máximo para la validez de un dato (*LIVELINESS*) o incluso el nivel de fiabilidad de una publicación (*RELIABILITY*).

2.2. Servicios multimedia

El aumento de las prestaciones en los sistemas informáticos, la aparición de nuevos algoritmos de codificación multimedia cada vez más eficientes, junto con la mejora permanente de las redes de comunicación y de los dispositivos de encaminamiento, han hecho posible el desarrollo y despliegue de servicios para el procesamiento de flujos multimedia.

Con independencia de sus singularidades, en este tipo de sistemas siempre se pueden identificar dos tipos de aplicaciones: las productoras frente las consumidoras de flujos multimedia.

Las primeras obtienen la información desde algún dispositivo capturador o bien de algún dispositivo de almacenamiento que suministra los flujos multimedia previamente capturados. Por su parte, las aplicaciones consumidoras reciben dichos flujos con el objetivo consumirlos localmente, o bien para analizarlos y/o procesarlos antes de enviarlos a otros nodos consumidores, asumiendo en este caso el papel de productores.

Tradicionalmente, la provisión de servicios multimedia se ha basado en interacciones tipo cliente/servidor en las que cualquier procesamiento y/o transformación del flujo multimedia se realiza mediante servidores perfectamente localizados. Un ejemplo relevante –aunque cada vez menos usado– es el estándar H.323 [86, 175], en el que se define –como se explicará más delante– una arquitectura centralizada basada en servidores (*e.g. gatekeepers*, *gateways* y Multipoint Control Unit (MCU))

para la provisión de servicios tales como la transcodificación o mezcla de flujos.

Esta aproximación presenta una serie de problemas estructurales obvios: dificultad para escalar por sobrecarga de los recursos del servidor, además poca robustez por la dependencia de un único punto en la provisión del servicio, así como la rigidez del servicio ofrecido debido al fuerte acoplamiento entre los consumidores y los productores de la información.

Las peculiaridades intrínsecas de los flujos multimedia hacen que el desarrollo de aplicaciones y la provisión de servicios distribuidos –para este tipo de información– no sea trivial [71, 169]

Concretamente, a continuación se identifican las características más relevantes de los flujos multimedia que dificultan el desarrollo de aplicaciones distribuidas:

Heterogeneidad. Un flujo multimedia puede contener información de muy diversa índole que puede representar texto, imágenes estáticas, voz, audio, animaciones o vídeo, entre otros. Cada uno de estos contenidos se caracteriza por una serie de requisitos que le son propios, además la variabilidad es todavía mayor si se consideran para un contenido dado las diferentes calidades –diferentes tasas de datos [140]– que pueden requerir los distintos escenarios de ejecución.

Volumen de datos. Cada tipo de contenido multimedia –y en particular, cada *codec*– tiene unos requerimientos de ancho de banda para su correcta transmisión que puede llegar a ser muy exigentes –especialmente para el vídeo–.

Diversidad de formatos. Cada tipo de contenido multimedia tiene sus propios formatos de almacenamiento y transmisión. El número de formatos y estándares existentes en la actualidad es muy elevado, siendo muchos de ellos incompatibles entre sí. Debido a este hecho, para poder comunicar tanto productores como consumidores deben estar acoplados, es decir deben usar los mismos formatos para poder intercambiar información.

Sensibilidad a retardos, a su variación y a pérdidas. Algunos flujos multimedia –por ejemplo de audio y vídeo– por la propia fisiología del usuario final son muy exigentes respecto al retardo extremo a extremo y a su variación o fluctuaciones. Además, dependiendo del diseño del *codec* y de la calidad demandada, pueden ser sensibles a las pérdidas.

En la última década han aparecido multitud de aplicaciones multimedia para realizar procesamiento distribuido de flujos multimedia [127]. En esta sección se realizará un recorrido sobre las aplicaciones multimedia distribuidas más significativas abarcando desde los inicios hasta las tendencias más recientes como *peer to peer* y *cloud computing*.

2.2.1. Perspectiva histórica

La aparición de computadores y dispositivos cada vez más potentes durante la década de los 90, hizo posible la aparición de sistemas multimedia en los que las aplicaciones productoras y consumidoras –por ejemplo sistemas de colaboración remota [141]– no se encontraban en el mismo nodo, sino que estaban localizadas en sistemas en red.

Es de destacar que a comienzos de la década de los 90, el transporte de contenido multimedia para la construcción de aplicaciones distribuidas era un reto aún abierto debido a las limitaciones de los sistemas y tecnologías disponibles. Dichas limitaciones, unidas a las restricciones debidas a los fuertes requerimientos impuestos por las características de los flujos multimedia, dificultaban enormemente el desarrollo de este tipo de aplicaciones. Así, en [190] el autor hace un análisis de los retos que existen a la hora de construir sistemas multimedia distribuidos.

Durante este periodo, la arquitectura cliente/servidor era la dominante en el despliegue de aplicaciones distribuidas. Uno de los sistemas de colaboración multimedia pionero fue MMConf [40]. Aunque ya existían contribuciones en este campo [172], MMConf fue el primero en proporcionar soporte para compartición de vídeo.

Otra propuesta relevante –también de principios de esta década– fue *Galatea Video Server* del MIT [9]. Galatea proporcionaba acceso a recursos remotos de vídeo mediante interacciones cliente servidor. Basado en este servicio, [143] muestra un caso de uso de un despliegue basado en *Galatea*.

Posteriormente, en [5] los autores proponen *BERKOM Multimedia Communication Service*, un servicio de colaboración orientado a entornos multiconferencia. Dicho proyecto delegaba el plano de transmisión multimedia al sistema *BERKOM Multimedia Transport System* [44]. Este sistema, se basaba en el protocolo Internet Stream Protocol (version 2) (SP-II) [178], protocolo obsoleto que puede ser considerado como uno de los precursores de Real-Time Protocol (RTP). Este trabajo es de los primeros en hacer uso de calidades de servicio (QoS) para construir sistemas multimedia distribuidos. Poco después, en [93] se propone PANDORA, una arquitectura orientada al manejo de flujos de audio y vídeo en tiempo real en redes Asynchronous Transfer Mode (ATM).

Dentro de toda revisión histórica sobre servicios multimedia distribuidos, merecen especial mención las especificaciones de la International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) y particularmente el conjunto de normas conocido como H.323.

La ITU-T especificó a mediados de la década de los 90 una serie de estándares enmarcados dentro de las familias *ITU-T H* y *ITU-T G* que supusieron un hito importante en este campo. Concretamente, se desarrollaron estándares para audio

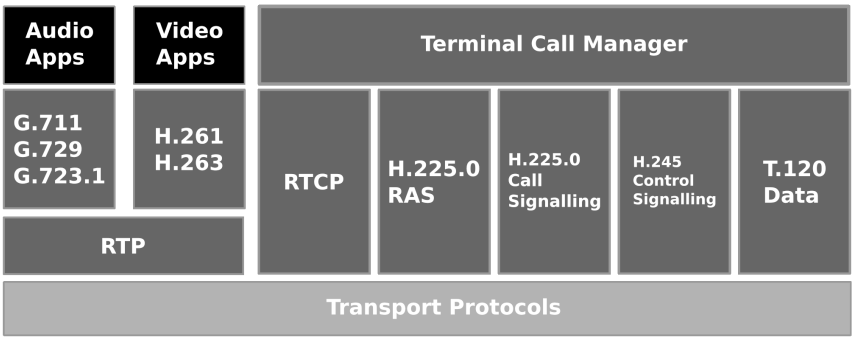


Figura 2.5: H.323: estándares y protocolos asociados

(serie *G.7XX*, por ejemplo *G.729* [84]), vídeo (serie *H.2XX*, siendo ejemplo significativos las normas *H.262* [85], *H.263* [87] y *H.264* [182]), para la señalización y gestión (serie *H.3XX*) y para texto *T.140* [88].

H.323 incluye un conjunto de normas específicas para el establecimiento, gestión y señalización de sesiones audiovisuales [86, 175]. Para un despliegue típico *H.323* –además de los terminales o puntos finales donde se generan o consumen los flujos multimedia– se definen las siguientes entidades (o servidores en red): *gateways*, *gatekeepers* y MCU.

Los primeros sirven de pasarela entre la red IP (adaptando los media y la señalización) y otro tipo de redes, como las de telefonía básica. Los *gatekeeper* son los encargados de control de admisión de llamadas y realiza otras tareas como registro de usuarios y resolución de direcciones.

Las MCU se encargan del control de multiconferencias, facilitando la combinación o mezcla de múltiples flujos generados desde distintos terminales *H.323* dentro de una misma sesión. Normalmente, las MCU se implementan en *hardware* dedicado (por lo general de alto coste), lo que limita la generalización de su uso.

Los despliegues basados en *H.323* permiten tanto la negociación de las características de los flujos multimedia a intercambiar como la adaptación y transformación de los mismos en caso de que sea necesario. El estándar *H.323* ha gozado de gran popularidad durante muchos años, pero paulatinamente ha sido sustituido por algunas especificaciones de la Internet Engineering Task Force (IETF) tales como RTP o Session Initiation Protocol (SIP) [151, 165]. No obstante, aunque cada vez en menor medida, se sigue usando en diferentes escenarios [197].

Finalmente, remarcar la importancia de las comunicaciones *multicast* (entre clientes y servidores) en los sistemas de distribución de contenido multimedia en tiempo real [109, 111]. Aunque especialmente indicadas cuando la cardinalidad entre servidores y clientes es de uno a muchos, esta aproximación no goza de gran popularidad

recientemente debido a que las comunicaciones *multicast* distan de estar habilitadas de forma generalizada en todas las redes y/o operadores. En este sentido, merece destacar la alternativa M-Bone (*Multicast Backbone*) que consistía en una infraestructura multicast a nivel de Internet para el establecimiento de sesiones multimedia. Sin embargo, la dificultad en el despliegue ha hecho que en la actualidad esté casi en desuso.

En la siguiente sección, motivados por las limitaciones y dificultades comentadas anteriormente y por su relación directa con los objetivos de la Tesis, se describen algunas propuestas para la distribución de contenido multimedia basadas en publicación/subscripción.

2.2.2. Multimedia y publicación/subscripción

Algunos escenarios para la distribución de contenido multimedia requieren modelos diferentes a la interacción cliente/servidor. En concreto, algunos autores han propuesto el uso de paradigmas de interacción más avanzados como el de publicación/subscripción [27, 43, 78]. En esta sección se llevará a cabo el estudio de algunas propuestas de distribución de contenido multimedia que usan este paradigma de comunicación.

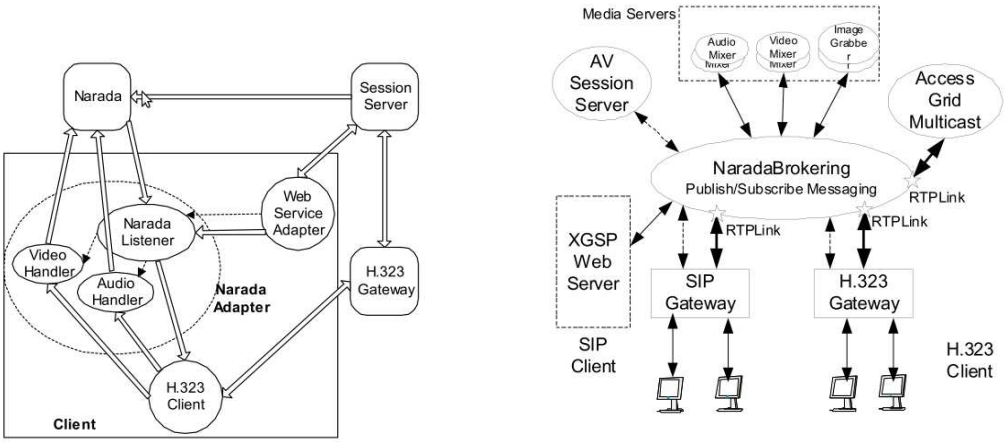
2.2.2.1. CORBA/NaradaBrokering

Una de las tecnologías de mayor relevancia durante finales de los 90 y principios de la siguiente fue Common Object Request Broker Architecture (CORBA), una tecnología sencilla para el desarrollo de sistemas distribuidos basados en componentes.

Dentro de esta categoría, cabe destacar por su carácter pionero a Toolkit for Open Adaptive Streaming Technologies (TOAST) [59], un *middleware* de distribución de contenido multimedia basado en componentes remotos que permite el desarrollo de aplicaciones multimedia adaptables. TOAST proporciona una serie de componentes que interactuan y que permiten distribuir y adaptar el contenido multimedia a los diferentes requisitos de los clientes interesados.

Otra solución basada en CORBA es [19, 60, 180]. En estos trabajos se propone una arquitectura de publicación/subscripción de contenido multimedia orientada a entornos de colaboración y videoconferencia, que hace uso de *NaradaBrokering* como sistema de comunicación [62].

NaradaBrokering es un servicio distribuido de distribución de eventos genéricos basado en *brokers*. Entre las características más relevantes del mismo, destacan:



(a) Arquitectura propuesta (b) Integración

Figura 2.6: Publicación de audio/vídeo basada en *NaradaBrokering*.

- Es un *broker* genérico, no diseñado específicamente para contenido multimedia.
- Soporta dos modos de comunicación: punto a punto (*peer-to-peer*) y publicación/subscripción.
- Ha sido desarrollado de acuerdo a la especificación JMS, por lo que es compatible a nivel de API con otras aplicaciones que cumplan con dicho estándar.

Como se ha mencionado anteriormente, a pesar del diseño inicial no específicamente concebido para servicios multimedia, posteriormente se han publicado varios trabajos para extender *NaradaBrokering* hacia el soporte de capacidades multimedia. Concretamente, en [19] se aborda por primera vez el tema de la integración de contenido multimedia con *NaradaBrokering*. En este trabajo se presentan los componentes principales para añadir soporte multimedia: *VideoHandler*, *AudioHandler*, y *Web Service Adapter* (Figura 2.6a).

Otro trabajo posterior [180] estudia el encapsulamiento de tráfico RTP dentro de paquetes de *NaradaBrokering*, así como la evaluación del impacto incurrido por el sistema. En este trabajo, se muestra que aunque la adopción de *NaradaBrokering* aumenta el retardo y el *jitter*, el incremento está dentro de los límites aceptables para una arquitectura distribuida multimedia en tiempo real.

Posteriormente, en [60] los autores presentan cómo un sistema multimedia basado en *NaradaBrokering* puede ser integrado con otras soluciones multimedia basadas en H.323 o SIP (Figura 2.6b).

Por último, en [61] se propone una arquitectura distribuida colaborativa para la anotación y análisis de eventos deportivos basada en el sistema de paso de mensajes

con publicación/subscripción y notificación de eventos de *NaradaBrokering*.

2.2.3. Sistemas *peer to peer*

Uno de los paradigmas arquitectónicos que ha tomado gran relevancia últimamente en sistemas distribuidos multimedia es el paradigma *peer to peer* basado en redes *overlay* [113]. En este paradigma, todos los nodos se organizan formando una red o recubrimiento a nivel de aplicación, de tal forma que se favorece la cooperación entre pares para por ejemplo repartir la carga entre todos los nodos que conforman el *overlay*. (Figura 2.7).

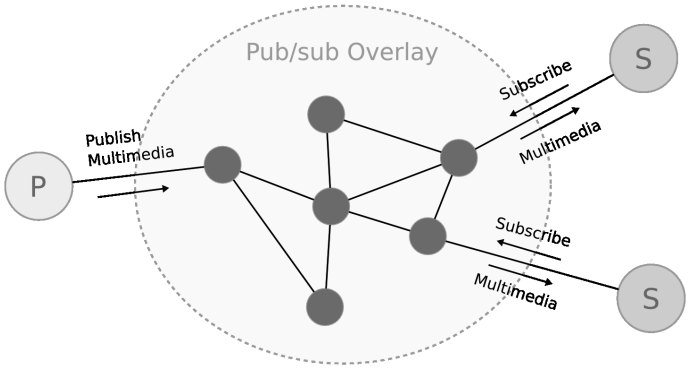


Figura 2.7: Red *peer to peer* para la distribución de contenido multimedia.

ChoPS [106] es un sistema de publicación/subscripción basado en redes *overlay* diseñado para transmisión y procesamiento de vídeo de manera distribuida. ChoPS usa una tabla *Hash* distribuida (DHT) para mantener la estructura del *overlay*. Concretamente, en ChoPS se propone el uso de Chord [173]. La principal aplicación de ChoPS es el procesamiento distribuido de flujos de vídeo. Actualmente, ChoPS se usa dentro del proyecto de videovigilancia distribuida Geoide PIV-17 [52].

En [177] los autores proponen Open Hypermedia System (OHS), un sistema de publicación/subscripción basado en ontologías para el descubrimiento de contenido multimedia enriquecido. Este sistema permite distribuir recursos multimedia que pueden ser localizados y mediante anotaciones. No obstante, este trabajo se centra en proporcionar un *framework* de alto nivel para crear un sistema de publicación/subscripción multimedia sobre un sistema *peer to peer*, no considerando las problemáticas asociadas con el transporte de datos multimedia.

Los autores de [78] proponen NovaPS, un sistema de publicación/suscripción para contenido multimedia basado en la construcción de una red *overlay*. Lo más relevante de este sistema es que propone el uso de filtros de Bloom [17] para el

emparejamiento entre las publicaciones existentes con los subscriptores.

Este tipo de arquitecturas son apropiadas para distribuir contenido multimedia en tiempo real cuando los requisitos de retardo y/o calidad no son determinantes, como por ejemplo la difusión de flujos de vídeo en tiempo real de eventos en directo (flujos unidireccionales no interactivos tales como conciertos o partidos de fútbol). Sin embargo, el mayor inconveniente que presentan las aproximaciones multimedia basadas en *peer to peer* es que debido a la cantidad de nodos intermedios que son necesarios para transferir un flujo de vídeo, normalmente la latencia introducida es grande y no se pueden garantizar calidades de servicio de extremo a extremo, haciendo así que no sean adecuados para sistemas con requisitos más estrictos como puede ser la vídeo vigilancia y/o sistemas interactivos.

2.2.4. Sistemas de *Cloud computing*

El reciente auge del *Cloud Computing* [1] ha propiciado al aparición de multitud de servicios multimedia distribuidos basados en esta tecnología [80, 200]. Esta tecnología permite alojar en *la nube* servicios para adaptar y transformar flujos multimedia para su distribución y consumo en dispositivos heterogéneos.

La aproximación *Cloud* está basada en el uso de recursos virtuales bajo demanda (*pay as you go*), lo que permite minimizar los costes fijos relacionados con la adquisición de *hardware* y el despliegue de los mismos.

Un ejemplo de servicios multimedia distribuidos en la nube se puede encontrar en [138]. En este trabajo se presenta un servicio de transcodificación de contenido multimedia en Internet para adaptarlo a las demandas específicas de dispositivos móviles, entre las que se pueden encontrar formatos, requerimientos de ancho de banda o resoluciones. Este sistema realiza una transcodificación en paralelo de vídeos disponibles en Internet (*e.g.* Youtube), logrando así reducir el tiempo de espera del usuario.

En [79] los autores proponen una arquitectura basada en *proxies* que permiten transcodificar contenido multimedia a formatos escalables de vídeo, permitiendo así al usuario seleccionar el nivel de calidad del flujo que desea recibir en base a las capacidades y requerimientos del dispositivo.

En [103] los autores proponen *i5Cloud*, un sistema híbrido que permite adaptar contenidos multimedia para dispositivos móviles. En este sistema se propone la utilización de recursos en la nube cuando la demanda de flujos multimedia supera un umbral que no puede ser satisfecho localmente. Así por ejemplo, si el número de flujos que deben ser transcodificados es muy grande, las peticiones que no puedan ser satisfechas se exportan a un servicio de *Cloud computing* externo (*offloading*). Este

sistema permite satisfacer los requerimientos de sistemas dinámicos manteniendo un coste de despliegue poco elevado.

En [97] se presenta XYLOMENOS, una arquitectura para dar soporte de publicación/subscripción en entornos móviles. En este trabajo se muestra un ejemplo de como la arquitectura PSIRP [149] de publicación/suscripción en Internet, puede ser empleada para la distribución de flujos multimedia.

2.2.5. Otros sistemas relevantes relacionados.

Además de los sistemas multimedia distribuidos mencionados anteriormente existen multitud de trabajos que por su relevancia para esta Tesis también merecen ser mencionados, pero que no pueden ser encuadrados en ninguna de las secciones anteriores.

Dentro de los sistemas para adaptar contenido multimedia de manera remota a dispositivos móviles se encuentra *Ambistream* [7]. Este sistema se centra en satisfacer los distintos requerimientos de formatos que presentan los dispositivos móviles más populares (*e.g.* Android o iPhone).

Relacionados con la aplicación de videovigilancia, existen una serie de trabajos que merecen ser mencionados. En primer lugar encontramos *TraffiCast*[201], un sistema de publicación/subscripción para videovigilancia en entornos híbridos *wireless*/redes de sensores. Otro trabajo relacionado es *Pub-Eye* [22, 202], un entorno de publicación/subscripción de videovigilancia sobre redes *wireless*. Ambos sistemas basan su funcionamiento en el uso de *brokers* intermedios para la distribución de contenido multimedia.

En [163] los autores proponen un *middleware* de publicación/subscripción para el desarrollo de aplicaciones multimedia distribuidas de colaboración remota. Asimismo, los autores demuestran cómo el sistema puede ser empleado para aplicaciones de pizarra compartida o incluso intercambio de flujos de vídeo entre participantes, para lo que proporcionan una API sencilla que habilita la comunicación entre nodos mediante el paradigma de publicación/subscripción basada en tópicos.

Más directamente relacionados con esta Tesis, existen una serie de trabajos que proponen el uso de tecnologías relacionadas con el estándar DDS para la distribución de contenido multimedia.

Así, en [4] los autores realizan un estudio de la viabilidad del uso de RTPS en sistemas de visión industrial. Sin embargo, este sistema únicamente se centra en la distribución de imágenes estáticas, no considerando flujos continuos (contenidos de audio y vídeo).

En [115] los autores hacen un análisis de las capacidades y potencial de DDS para la transmisión de contenido multimedia. Concretamente, los autores se centran en un estudio de las capacidades de calidad de servicio QoS que el estándar DDS proporciona y cómo pueden ser aplicadas a la distribución de contenido multimedia.

En [65] se propone el uso de DDS para el intercambio de flujos de vídeo entre nodos. Concretamente, este sistema se centra en el uso de DDS para el intercambio de flujos de vídeo en entornos de supervisión industrial. Para ello se propone la construcción de pasarelas entre flujos Real-Time Streaming Protocol (RTSP)/RTP y tópicos DDS.

En línea con el anterior trabajo, es destacable [45]. En este trabajo, los autores proponen el uso de DDS para realizar distribución de flujos de vídeo en entornos *wireless*. EN concreto se propone una arquitectura que permite la distribución de flujos de vídeo escalables (H.264/Scalable Video Coding (SVC)) [166] permitiendo así el uso de dispositivos con capacidades heterogéneas de forma simultánea. No obstante, tanto este sistema como el anterior no proporcionan soporte a otros tipos de medios (*e.g. audio*), ni proporcionan una arquitectura completa orientada a la transformación y procesamiento de flujos multimedia en tiempo real ya que se centran únicamente en la difusión de los flujos de vídeo.

2.2.6. Resumen.

En esta sección se ha realizado un estudio del estado del arte de sistemas de distribución de contenido multimedia basados en el paradigma de publicación/subscripción. Entre los sistemas estudiados se han identificado sistemas orientados a la distribución de contenido multimedia a gran escala [78, 106, 177], sistemas orientados a la videovigilancia [22, 201, 202], sistemas colaborativos [19, 60, 61, 163, 180], sistemas orientados al control de procesos industriales [4, 65] y sistemas orientados a la transformación de contenido multimedia para ser adaptado a los requerimientos de diversos dispositivos [79, 103, 138].

Como se puede observar, cada uno de estos trabajos se centra en un aspecto determinado de la distribución de contenido multimedia o se centra en un tipo de media determinado (*e.g.* vídeo), no existiendo ninguna solución integradora que permita ser utilizada en múltiples escenarios.

En el Capítulo 4, y tomando como partida las ventajas e inconvenientes de los sistemas identificados, se propondrá un marco de trabajo que puede satisfacer de forma conjunta los requisitos de los escenarios comentados anteriormente usando una aproximación de publicación/subscripción centrada en datos.

2.3. Monitorización de recursos en entornos distribuidos

Con la aparición de los primeros sistemas distribuidos se hizo patente la necesidad de monitorizar el estado y uso de los recursos disponibles en dicho sistema. En esta sección se revisan las contribuciones más relevantes en este campo para con ello establecer un punto de partida y situar en contexto la investigación realizada.

Para una mejor comprensión, esta sección se ha organizado del siguiente modo: en la sección 2.3.1 se revisan los antecedentes históricos en monitorización de dispositivos en red. Seguidamente, en la sección 2.3.2 se propone una taxonomía de los sistemas de monitorización que permite la categorización de los sistemas presentes en el estado del arte. Finalmente, en las secciones 2.3.3, 2.3.4, 2.3.5 y 2.3.6 se estudian los sistemas de monitorización más relevantes.

2.3.1. Antecedentes históricos

La monitorización de sistemas de computadores ha evolucionado considerablemente a lo largo de la historia. De hecho, en la actualidad está tomando una relevancia cada vez mayor debido a una mayor implantación de sistemas distribuidos y especialmente en los últimos años con el auge de tecnologías como el *Cloud Computing* [20].

Uno de los primeros estándares para la monitorización de dispositivos en red fue el conocido como Host Monitoring Protocol (HMP) (especificado en el documento IEN 197 que posteriormente evolucionó al RFC 869 [76]). Este estándar –en la actualidad obsoleto– definía un protocolo de intercambio de información de monitorización entre todo tipo de dispositivos de red (*hosts*, *gateways*, *routers*) con un servidor central llamado *monitoring center*. Este sistema centralizado asumía que la capacidad de procesamiento reside en el *monitoring center*, y por tanto los dispositivos se limitan a recolectar estadísticas de uso de recursos primitivas y transmitirlas al *monitoring center*.

En esta especificación se observan una serie de características que aún perduran en muchos sistemas actuales de monitorización: en primer lugar hace uso de un esquema petición/respuesta para el intercambio de información, en segundo lugar permite la notificación de eventos asíncronamente sin petición previa por parte del *monitoring center*, y en último lugar se permiten por parte de los servidores el envío de comandos de control a los *hosts* (*e.g.* cambio en la frecuencia de recolección de estadísticas).

Como respuesta a la aparición de nuevas necesidades y a la evolución de los

sistemas de computadores y las redes de comunicación, se hizo necesario el uso de nuevos estándares para la gestión de redes. Tal es el caso de Simple Network Management Protocol (SNMP) [25]. SNMP se definió originalmente como un protocolo para la gestión de redes, pero su uso más habitual ha sido el de monitorización [121, 123, 174, 189].

El funcionamiento de SNMP para monitorización es en cierto modo similar a HMP, ya que presenta características similares: está basado en interacciones petición/respuesta, señalización asíncrona en base a *traps*, y usa agentes recolectores de información llamados Network Management System (NMS). Par ello el estándar SNMP define una serie de operaciones (*e.g. GetRequest*, *SetResponse*) para el intercambio de información entre agentes y los NMS.

A lo largo de los años, SNMP ha evolucionado en sucesivas versiones tratando de satisfacer nuevos requerimientos. Ejemplo de estas evoluciones son SNMPv2 [24] y SNMPv3 [74], que alcanzó en 2002 el nivel *full standard* de la IETF.

Tanto HMP como SNMP definieron algunas de las capacidades más relevantes que debe tener cualquier sistema de monitorización. No obstante, aunque son una solución válida para la monitorización de muchos escenarios, sus características imponen limitaciones, por lo que no son adecuados para todos casos, como por ejemplo para la monitorización en tiempo real de sistemas multimedia. En particular, SNMP está orientado a la monitorización de redes (*e.g. routers* y *switches*) y se basa principalmente en un esquema de petición/respuesta, no siendo por tanto apropiado para sistemas dinámicos que requieran actualización en tiempo real.

2.3.2. Clasificación de los sistemas de monitorización

Para una mejor comprensión del estado del arte en sistemas de monitorización, en este apartado ser realiza una clasificación de los mismos en base a una serie de características básicas que pueden presentar.

En primer lugar se considera la arquitectura empleada, pudiéndose distinguir dos tipos de sistemas: centralizados o distribuidos.

La segunda característica para llevar a cabo su clasificación es el modelo de comunicación empleado, o lo que es lo mismo, qué nodo es el responsable de iniciar la comunicación entre los nodos.

Finalmente, para la clasificación aquí descrita se tiene en cuenta el modelo de despliegue, en función de si se utilizan programas específicos para realizar la recolección de datos de monitorización, o sin embargo se realizan mediante sondeos usando herramientas de propósito general. A continuación y a lo largo de este apartado se procederá a describir cada una de estas categorías de clasificación.

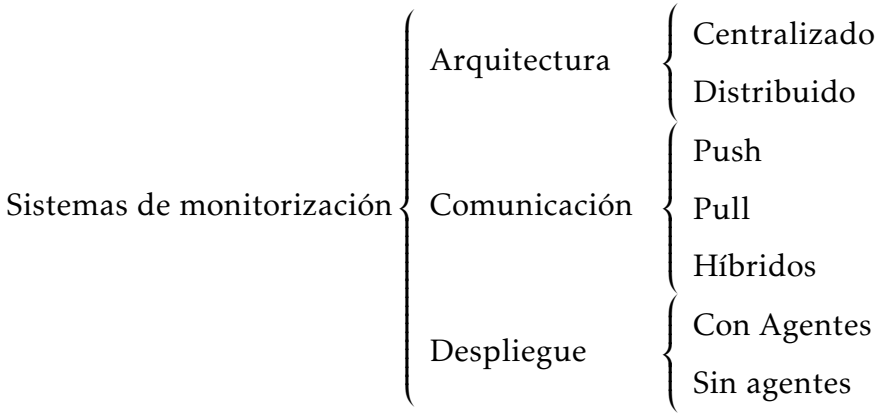


Figura 2.8: Clasificación de los sistemas de monitorización

Centralizado *vs*. Distribuido. Una primera metodología para categorizar los sistemas de monitorización se basa en el modelo arquitectónico empleado. Teniendo en cuenta dicho modelo, se pueden observar dos tipos de sistemas: centralizados *versus* distribuidos.

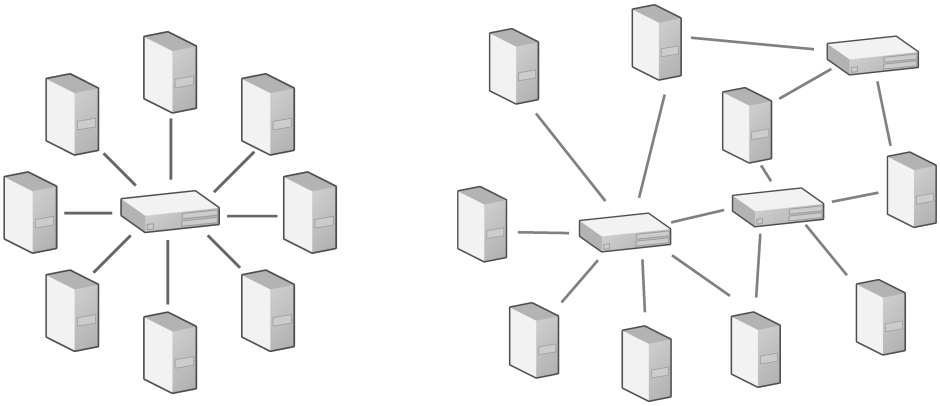
En el primer caso una entidad central es la encargada de almacenar toda la información de monitorización del sistema, mientras que en el segundo la información de monitorización se encuentra distribuida en toda la red.

Los sistemas centralizados presentan una serie de problemas bien conocidos –como la escalabilidad y robustez [94]– que han sido ampliamente discutidos y debatidos en la bibliografía [116].

Como se menciona anteriormente, los sistemas de monitorización centralizados almacenan toda la información del estado del sistema en una única base de datos, la cual es consultada cada vez sea necesario. Debido a esto, una aproximación centralizada puede suponer un cuello de botella cuando el flujo de información de monitorización presente en el sistema sea grande. Esto ocurre por ejemplo cuando el número de dispositivos a monitorizar o la frecuencia de actualización del uso de los recursos en el sistema sean elevados. En ambos casos, la sobrecarga de la base de datos puede llegar a saturar al servidor de monitorización.

Hay que tener en cuenta además que es posible que haya muchas entidades interesadas en consultar el estado de los recursos, por lo que la tasa de peticiones de estado de recursos puede aumentar, llegando fácilmente a la saturación del sistema.

También se ha mencionado el problema de la robustez, y es que al existir un único punto de almacenamiento de información, el sistema es más vulnerable: si dicho nodo falla, la estabilidad de las aplicaciones dependientes de la información



(a) Centralizado (b) Distribuido

Figura 2.9: Centralizado vs. distribuido

de monitorización puede verse en entredicho.

Por otra parte, las aproximaciones centralizadas tienen como ventaja una mayor facilidad de implementación. Un ejemplo relevante de monitorización centralizada es *Nagios* [124].

Para solventar los problemas comentados en los sistemas centralizados, surgen los sistemas de monitorización distribuidos. En estos el reparto de la carga tiene como ventaja la posibilidad de organizar el sistema monitorizado de diversas formas, como por ejemplo adoptando organizaciones jerárquicas o federadas.

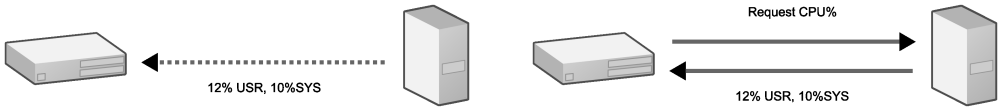
En las aproximaciones distribuidas, cada nodo se encarga de almacenar únicamente la información de monitorización de un subconjunto del total de recursos del sistema, resultando por tanto en una menor carga de actualización, mejorando así la escalabilidad del sistema.

Además, en caso de fallo de uno de los nodos de almacenamiento, solo la parte del sistema dependiente del mismo se vería afectada, resultando pues en una mayor robustez total del sistema.

Algunos ejemplos de sistemas de monitorización distribuidos pueden encontrarse en [116, 154, 196]

Push vs. Pull. Otra posible clasificación para los sistemas de monitorización es atendiendo al modelo de distribución de datos adoptado. En este sentido se observan dos tipos de estrategias principales: *pull* [121, 124] o *push* [116].

En el modelo *pull*, la entidad interesada en recolectar la información de moni-



(a) Push (b) Pull

Figura 2.10: Pull vs. Push

torización es la que realiza peticiones de estado a los dispositivos monitorizados. Dichas peticiones pueden ser periódicas o bajo demanda. La ventaja de estos sistemas es que cada una de las entidades interesadas puede establecer la frecuencia a la que desea recibir actualizaciones. Como contrapartida, estos sistemas presentan una mayor latencia, ya que requieren la realización de una petición y la recepción de la correspondiente respuesta para obtener datos de monitorización actualizados.

Como contrapartida, en en el modelo *push*, es la entidad monitorizada la que envía las actualizaciones de estado a las entidades interesadas. Al igual que en el modelo *pull*, en este modelo también existen dos aproximaciones al enviar las actualizaciones: periódica o dirigida a eventos.

La ventaja de la actualización periódica es que permite tener la información actualizada y precisa de toda la red, pero el precio a pagar es un mayor uso de ancho de banda para distribuir dicha información.

Por otro lado, la aproximación basada en eventos, permite definir una serie de reglas que disparen el envío de actualizaciones a las entidades interesadas (*e.g.* alcanzar un nivel de uso de recursos determinado). Esta aproximación consigue por lo tanto un uso más eficiente de recursos, ya que solo envía por red actualizaciones en determinadas circunstancias. A cambio, la entidad monitorizadora únicamente dispone de una estimación aproximada del uso actual de un recurso.

Este hecho hace de la aproximación basada en eventos más propicia en escenarios donde más que el uso exacto de recursos, prime la necesidad de identificar eventos para ejecutar acciones (*e.g.* alarmas).

***Agent-based, Agent-less* o basados en *scripts*.** Finalmente, una última clasificación para los sistemas de monitorización se basa en el modelo adoptado en la recolección de datos. Teniendo en cuenta este aspecto se identifican tres tipos de sistemas, cada uno con sus ventajas e inconvenientes: basados en agentes [116], sin agentes [198] y basado en *scripts* [63].

En el caso de los sistemas basados en agentes, cada nodo que está siendo monitorizado debe instalar un software dedicado a recolectar y enviar datos de monitoriza-

ción del nodo (agente). La ventaja de estos sistemas es que permiten al usuario/administrador personalizar enormemente las métricas que pueden ser recolectadas de cada nodo. Por otro lado, el mayor inconveniente que presenta esta aproximación es que el coste de despliegue es mucho mayor, ya que los agentes deben ser instalados y tal vez configurados en cada uno de los nodos a monitorizar.

Como contrapartida, los sistemas sin agente son aquellos que no necesitan instalar en cada nodo monitorizado un software dedicado para la recolección y difusión de datos de monitorización. Normalmente, este tipo de sistemas solo permiten recolectar aquellas métricas que son visibles desde el exterior del sistema, no teniendo por lo tanto acceso a métricas de bajo nivel como pueden ser por ejemplo el uso de recursos hardware (CPU, memoria, etc). Este tipo de sistemas por tanto es más utilizado para recolectar métricas pasivas de alto nivel, como por ejemplo la conectividad con un nodo, la disponibilidad de un servicio (*e.g.* servidor HTTP), o el *round-trip-time* hacia ese nodo en concreto.

Existe además un tipo de despliegue intermedio basado en *scripts*. Éste consiste en la ejecución de programas de recolección de datos en cada uno de los *hosts* a monitorizar. Dichos *scripts* son ejecutados desde otros servicios no dedicados como *telnet*, Secure SHell (SSH) o Common Gateway Interface (CGI) y devuelven la información de uso en un determinado formato estructurado, como por ejemplo Comma Separated Values (CSV), JavaScript Object Notation (JSON) [39] o XML [16]. Si bien este sistema puede obtener una flexibilidad similar a los sistemas basados en agentes, presenta inconvenientes de seguridad, ya que habilitar esos servicios para poder ejecutar *scripts* suele requerir acceso al sistema con privilegios de ejecución, lo que supone una vulnerabilidad. Un ejemplo de este tipo de despliegue es Nagios Remote Plugin Extension (NRPE) [63].

2.3.3. Nagios

Uno de los sistemas más populares de monitorización de recursos es *Nagios* [14, 82, 117, 124] (anteriormente conocido como *Netsaint*). *Nagios* es un sistema centralizado de monitorización de recursos en la red (Figura 2.11).

Por defecto, *Nagios* está orientado a la monitorización de recursos y métricas pasivas de alto nivel, tales como conectividad con *hosts* y accesibilidad a determinados servicios. Además, *Nagios* proporciona por defecto acceso a los datos de monitorización por medio de una interfaz *web*, por lo que está más orientado a ser usado como herramienta de consulta y visualización de información de monitorización que como herramienta de gestión automatizada.

Una de las características que ha hecho que *Nagios* goce de gran popularidad es su capacidad de añadir fácilmente funcionalidades. Gracias a esta capacidad los

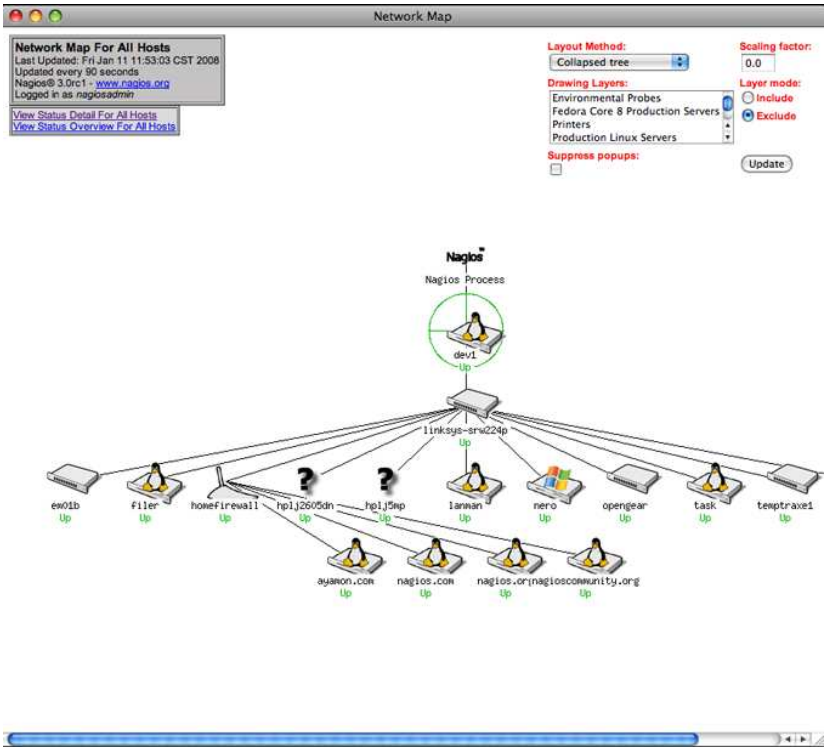


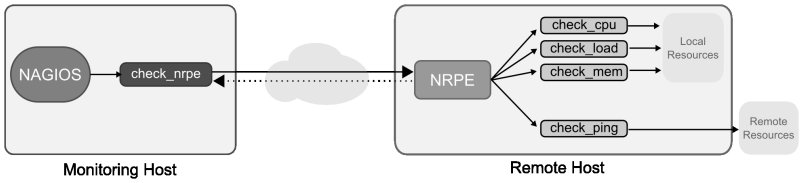
Figura 2.11: Captura de Nagios

administradores pueden desarrollar sus propias métricas e integrarlas dentro de la interfaz de *Nagios* fácilmente.

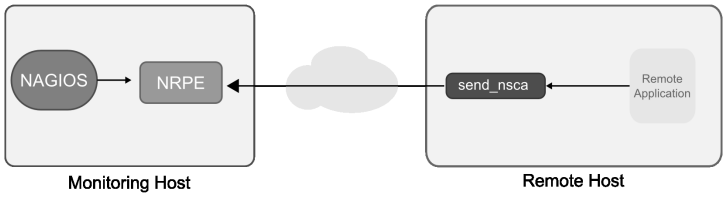
De este modo, *Nagios* se puede adaptar a múltiples escenarios gracias a su capacidad de incorporar nuevas funcionalidades mediante el uso de extensiones (*addons* y *plugins*) [126]. El abanico de extensiones disponibles es amplio y van desde el soporte de nuevos servicios –como especializaciones orientadas a *Cloud Computing*– hasta nuevos mecanismos de notificación a administradores.

Por su relación con el objetivo de la Tesis, se destacan dos extensiones especialmente relevantes: NRPE (Figura 2.12a) y NSCA (Figura 2.12b).

La primera extensión, NRPE, está orientada a la ejecución de *scripts* remotos para obtener mediciones de recursos locales que no pueden ser obtenidos con los mecanismos estándar que proporciona Nagios (*e.g.* uso de CPU, memoria, etc). Para ello se instala un agente NRPE en cada *host* a monitorizar. Dicho agente es encargado de ejecutar bajo petición del servidor central un conjunto de *scripts* que devuelven en texto plano la salida de los *scripts* (Figura 2.12a). El servidor de Nagios es encargado entonces de almacenar dicha información en la base de datos local. Como puede observarse este mecanismo es bastante sencillo, lo que permite crear extensiones fácilmente.



(a) Nagios NRPE



(b) Nagios NSCA

Figura 2.12: Extensiones de Nagios: NRPE y NSCA.

Sin embargo, presenta una serie de problemas: el primero de ellos es que el esquema petición/respuesta introducido por NRPE supone una latencia mayor debido a que hay que añadir al tiempo de transmisión necesario para la petición y la respuesta, el tiempo de ejecución del *script* de recolección de datos (Ecuación (2.1)). Si bien para la mayoría de los recursos este tiempo es pequeño, para algunos recursos que requieren realizar varias mediciones en un intervalo de tiempo para su estimación (*e.g.* CPU, uso de red, etc), el retardo incurrido puede ser grande.

$$T_{retardo} = T_{pet} + T_{script} + T_{resp} \tag{2.1}$$

El segundo inconveniente está más relacionado con el propio diseño de *Nagios*, y es que *Nagios* no está diseñado para realizar mediciones con frecuencias elevadas (con periodos del orden de segundos o menores). Por lo tanto, no es apropiado para escenarios que requieran estimaciones exactas y relativamente frecuentes del uso de recursos como es el caso de escenarios multimedia.

La otra extensión de interés de *Nagios* es NSCA (Figura 2.12b). Esta extensión está orientada a realizar notificaciones pasivas al servidor central de *Nagios*. El cometido inicial de esta extensión es comprobar localmente el correcto funcionamiento de los servicios y la disponibilidad de los recursos, para posteriormente enviar notificaciones al servidor central de *Nagios*. De este modo, los despliegues basados en NSCA son distintos a los empleados en NRPE: en este caso se instala una utilidad que se invoca desde la línea de comandos que recibe como argumentos el estado de un recurso o servicio determinado. Dicha información de estado es empaquetada y

enviada al servidor *Nagios*, que es el responsable de su almacenamiento en la base de datos local.

Aunque esta aproximación permite solventar las carencias de NRPE, presenta otros inconvenientes que justifican la investigación de otras alternativas orientadas conseguir los objetivos que esta Tesis persigue. Uno de estos inconvenientes es que NSCA usa TCP como protocolo de transporte. Esto conlleva que cada vez que se envía una notificación al servidor es necesario establecer una conexión para después cerrarla. En escenarios grandes –o con tasas de actualización muy altas– esto puede implicar una sobrecarga muy grande de la red debido a que el número de conexiones establecidas por segundo puede llegar a ser muy elevado.

Resumiendo, *Nagios* es uno de los sistemas de monitorización de *data-centers* con mayor difusión [125]. Su facilidad de instalación y su interfaz web lo hace un candidato ideal para muchos escenarios. Sin embargo, algunas de sus características dificultan su adopción en entornos cambiantes o donde sea deseable una monitorización con requisitos de tiempo-real. Ejemplos de estas características son su arquitectura centralizada y la necesidad de ser configurado manualmente mediante ficheros.

2.3.4. Ganglia

Ganglia [64] es un proyecto de la Universidad de California, Berkeley (USA) que surge ante la necesidad de tener un sistema capaz de monitorizar entornos de High Performance Computing (HPC) tales como *clusters* de computadores, de una manera escalable y confiable. Actualmente se encuentra financiado y mantenido por parte del proyecto *Planet Lab* [144].

La estructura jerárquica de los componentes de *Ganglia* lo hace un candidato idóneo para la monitorización de entornos *cluster* federados o incluso en *Grid Computing*.

Una infraestructura basada en *Ganglia* se compone de dos procesos residentes (*daemon*) principales que se organizan jerárquicamente: *gmond* y *gmetad*.

En el nivel inferior (nivel de *cluster*), *gmond* se despliega en cada nodo a monitorizar y es el responsable de recopilar el uso de recursos de ese nodo así como de publicarlos en el entorno del cluster por medio de mensajes de multidifusión en formato eXternal Data Representation (XDR) [51].

En un nivel superior se despliega el *daemon gmetad*, encargado de recolectar y agregar la información de una o varias federaciones de *clusters*. El intercambio de datos se realiza en este nivel mediante mensajes XML usando conexiones TCP y un esquema petición/respuesta. Dichos mensajes contienen la agregación de toda la

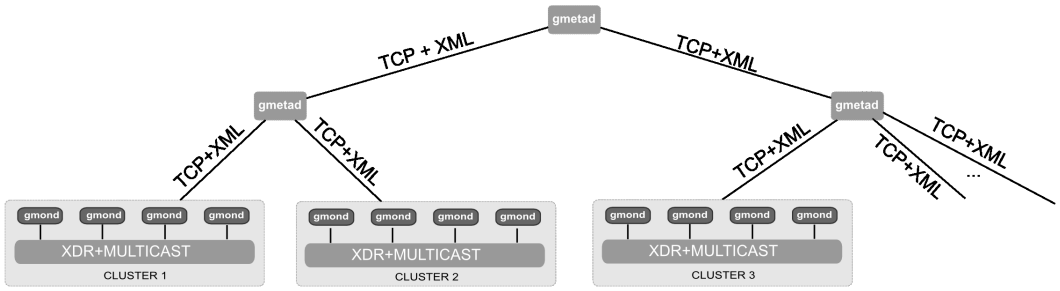


Figura 2.13: Arquitectura de *Ganglia*.

información de monitorización disponible en el *cluster*. El *daemon gmetad* se puede situar en niveles superiores de la jerarquía, pudiendo así monitorizar de manera escalable entornos con un gran número de nodos [144]. La Figura 2.13 muestra la arquitectura propuesta por *Ganglia*.

Ganglia presenta algunas características deseables en sistemas de monitorización que otros competidores como *Nagios* no presentan. Por su relevancia, es destacable el uso de un procedimiento de descubrimiento automático a nivel *intra-cluster*. *Ganglia* en este aspecto utiliza un protocolo basado en anuncio/escucha que permite añadir nodos a un *cluster* sin necesidad de configuración manual. Sin embargo hay algunos aspectos para los que no ofrece solución, como por ejemplo la administración remota de nodos, el descubrimiento a nivel *inter-cluster* o el soporte de alarmas.

2.3.5. Lattice

Otro de los sistemas de monitorización recientes con amplia difusión es *Lattice* [29, 31, 32]. *Lattice* surge dentro del proyecto europeo *RESERVOIR* [157] para satisfacer las necesidades de monitorización de recursos en entornos *Cloud Computing*.

Una de las características más relevantes de este sistema –en comparación con otros sistemas de monitorización– es el esquema de comunicación empleado: mientras que muchos sistemas emplean la aproximación petición/respuesta, *Lattice* adopta un paradigma de publicación/subscripción.

Para ello construye una arquitectura basada en entidades fuentes (*DataSources*) y consumidoras (*Consumer*). De acuerdo a este modelo, los productores instalan una serie de sondas (*Probes)* en los sistemas a monitorizar. Cada uno de estas sondas, está encargada de recolectar el estado de un determinado recurso, ya sea físico o virtual. Una vez recolectada dicha información, el *DataSource* al que se asocia la

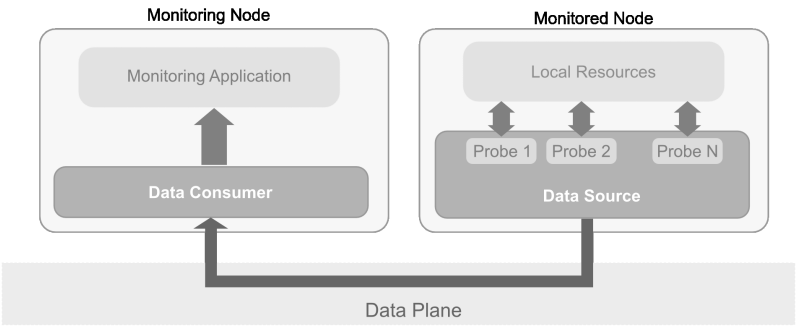


Figura 2.14: Arquitectura de Lattice.

sonda la publica haciéndola llegar a los *DataConsumer* asociados. En la Figura 2.14 se pueden observar los componentes de *Lattice* y las relaciones entre ellos.

Cuando se compara con otros sistemas de monitorización, *Lattice* presenta algunas novedades que lo hacen más adecuado para escenarios grandes y altamente cambiantes. Una de las más destacables es que ofrece mecanismos para establecer filtros de publicación, evitando saturar innecesariamente a los nodos que no necesitan una actualización constante acerca del uso de recursos.

2.3.6. Otros sistemas de monitorización relevantes

Además de los sistemas de monitorización anteriormente descritos, existen otros sistemas –que aunque no disfrutan de tanta difusión como los anteriores– necesitan ser tenidos en cuenta.

Uno de ellos es *MonALISA* [129], un sistema distribuido multi-agente orientado a monitorizar entornos *Grid*. MonALISA está construido por un conjunto de servicios basados en tecnologías JINI/JAVA que se comunican mediante Web Services Description Language (WSDL)/Simple Object Access Protocol (SOAP)[41]. Para el descubrimiento de nuevos servicios, *MonALISA* usa un registro centralizado, mientras que para la recolección de datos usa protocolos como SNMP.

Los autores de *SMon* abordan el problema de la distribución de datos de monitorización en [196]. La solución propuesta está orientada a reducir el retardo de monitorización mediante el mantenimiento de una red *overlay* optimizada para conseguir una reducción del retardo y la sobrecarga ocasionada en al red.

Una propuesta de integración de tecnologías de código abierto para la monitorización de entornos *Cloud Computing* privados es *PCMONS* [42]. En este esquema, la propuesta inicial hecha por los autores se basa en la utilización de *Nagios* como uno de los componentes principales de monitorización, con lo que esta solución presenta

los mismos problemas que éste.

Grid Resource Information Monitoring (GRIM) es un sistema orientado a monitorizar recursos en entornos *Grid* [28]. Esta propuesta se centra en dos puntos relevantes: la primera consiste en usar una aproximación *push* para la distribución de datos y la segunda se centra en el desarrollo de un conjunto de algoritmos de distribución de datos que minimicen el número de mensajes enviados. Como contrapartida, este trabajo no estudia cuestiones importantes tales como describir la arquitectura ni hace uso de tecnologías estandarizadas.

Otra propuesta interesante en esta dirección es P&P [77]. En este caso se propone una aproximación híbrida entre *pull* y *push* para la difusión de información. Sin embargo, este sistema se limita a proponer un algoritmo para reducir el número de peticiones, no definiendo el resto de la arquitectura del sistema de monitorización

Dentro de las soluciones comerciales está Hyperic HQ [187], una solución *open source* de monitorización desarrollada por VMWare Inc.[186]. Hyperic HQ utiliza una aproximación basada en agentes que descubren automáticamente los recursos locales. No obstante, la aproximación utilizada es centralizada –pues toda esta información es enviada a un servidor central– siendo, por tanto, vulnerable a los problemas comunes a los sistemas que adoptan esta arquitectura.

En [75] se propone una arquitectura de monitorización para *Cloud Computing* basada en el uso de un bus de datos común para el intercambio de información de monitorización. Este sistema propone el uso de un motor de agregación y filtrado de la información de monitorización para conseguir escalabilidad, sin embargo no hace mención a los requerimientos de calidad de servicio en la difusión de dicha información. Similarmente, en [102] se propone una solución para monitorización en *Cloud computing* basada en P2P para la distribución de los datos de monitorización recolectada mediante herramientas como Ganglia o Nagios. Ninguno de los dos sistemas anteriores, tiene en cuenta el uso de tecnologías estandarizadas para la difusión de datos, hecho que dificulta la integración con otras herramientas.

2.3.7. Comparación cualitativa entre sistemas de monitorización

Para concluir nuestro estudio, en la Tabla 2.3 se proporciona una comparación directa entre los sistemas de monitorización aquí presentados teniendo en cuenta las características que consideramos destacables para un escenario tan dinámico como es la distribución de servicios multimedia:

Arquitectura. En este caso se compara el paradigma de comunicación empleado.

Modelo de interacción En este ítem se caracteriza si los datos de monitorización se

	Arquitectura	Interacción	Descubrimiento	Filtrado	Formato	Actualizaciones
Nagios NSCA	C/S	Push	No	-	Texto plano	-
Nagios NRPE	C/S	Pull	No	-	Texto plano	Periódica
Ganglia [116]	Agentes	Push/Pull	Intra Cluster	-	XDR + XML	Periódico
Hyperic HQ [187]	Agentes	Pull	Solo sensores	-	Propietario	Periódica
PCMONS [42]	Agentes	Push	No	-	Texto plano	Periódica
Lattice [32]	P/S	Push	Si (Multicast)	Tiempo	XDR	Periódica y Eventos
[75]	P/S	Push	-	Si	Propietario	Periódica y Eventos
[102]	P/S	Push	No	Si	Propietario	Periódico

Tabla 2.3: Comparación directa entre sistemas de monitorización de recursos

envían bajo un esquema de petición/respuesta (*pull*), bien se generan y posteriormente transmiten desde la fuente al monitorizador.

Descubrimiento. En este concepto se caracteriza si el descubrimiento es manual o automático.

Filtrado. En algunos escenarios, los nodos interesados en la información de monitorización están interesados en recibir la totalidad de las actualizaciones de estado de los nodos monitorizados, o por el contrario solo necesitan recibir un subconjunto de estas de acuerdo a algún criterio. En este apartado analizamos el soporte de filtrado de los distintos sistemas de monitorización.

Formato. El formato empleado a la hora de intercambiar los datos de monitorización también es importante. El uso de formatos estructurados facilita el procesamiento y análisis de los datos. Por otro lado, los formatos estandarizados permiten una mejor integración con herramientas de terceros tales como bases de datos.

Política de actualización. De acuerdo a la estrategia de recolección y distribución de mediciones de recursos se pueden identificar dos posibilidades en los sistemas de monitorización: recolección periódica y basada en eventos. Mientras la primera la primera estrategia toma mediciones a intervalos fijos predefinidos, la segunda se basa en la aparición de algún evento relacionado o no con algún recurso (*e.g.* un recurso excede cierto umbral de uso).

3

Una arquitectura extensible para servicios multimedia

Construir sistemas para la provisión de servicios multimedia distribuidos de tiempo real puede llegar a ser una tarea compleja. Los flujos multimedia son especialmente sensibles a los retardos, a sus variaciones (*jitter*) y en cierta medida a los errores de transmisión. Un paquete no recibido a tiempo puede generar degradaciones en la calidad de la señal multimedia recibida.

Mientras que en algunas aplicaciones esto no es especialmente relevante –como por ejemplo las dedicadas al ocio– en otras aplicaciones –por ejemplo aquellas dedicadas operaciones críticas, como la vídeo-vigilancia, telemedicina, etc– estas degradaciones pueden ser inadmisibles. Esto ocasiona que el diseño de estos sistemas ha de estar especialmente cuidado para mantener la calidad esperada.

Otro hecho a tener en cuenta es que algunos entornos no requieren únicamente la transmisión de vídeo para su visionado o almacenamiento remoto, sino que además deben permitir de una forma eficaz su transformación, procesado o análisis. En definitiva se trata de entornos que necesitan enriquecer el servicio prestado permitiendo para la señal transmitida su etiquetado, seguimiento, anotación, sincronización, adaptación (transcodificación), etc.

Un ejemplo de esta necesidad en un entorno de vídeo-vigilancia sería poder –sobre el servicio básico de transmisión de la señal de vídeo– identificar objetos y en su caso hacer un seguimiento del movimiento y/o disparo de alertas por detección de anomalías. Otro ejemplo ilustrativo podría ser en una aplicación de vigilancia ambiental, la provisión de un servicio de vídeo enriquecido con animaciones superpuestas en base a la telemetría -en tiempo real- de los sensores de una red sobre el *stream* de audio/vídeo recogido desde un UAV.

Tradicionalmente estos sistemas han sido implementados adoptando arquitectu-

ras centralizadas que utilizan el paradigma cliente/servidor. En este caso, un servidor central es el encargado de recibir toda la información, hacer los procesamientos necesarios y distribuir posteriormente entre los clientes los distintos flujos resultantes mediante interacciones solicitud/respuesta.

A pesar de ello, como se explicó en el Capítulo 2, el paradigma cliente/servidor puede implicar una serie de problemas derivados de su arquitectura centralizada, como por ejemplo la falta de robustez ante posibles fallos del punto central o la falta de prestaciones (escalabilidad) por la aparición de cuellos de botella. Además, esta aproximación implica una falta de flexibilidad (o rigidez) en la provisión o despliegue de servicios debido al fuerte acoplamiento (espacial, temporal y en la lógica de la aplicación) que toda interacción cliente/servidor conlleva.

En este capítulo se propone una arquitectura para distribución y provisión de servicios multimedia. Dicha arquitectura permite la construcción de sistemas distribuidos que satisfagan múltiples escenarios multimedia con los requisitos más exigentes de forma eficaz, robusta, flexible y escalable.

Con este objetivo, el capítulo se ha organizado de la siguiente manera. En primer lugar, en la Sección 3.1 se realizará un análisis de los requisitos que debe cumplir el sistema a diseñar. Seguidamente en la Sección 3.2 se explicará el diseño de la arquitectura propuesta. En la Sección 3.3 se darán los detalles de implementación más relevantes. En la Sección 3.4 se abordará la construcción de servicios basados en EMDS. Seguidamente, en la Sección 3.5 se explica el uso de EMDS en entornos de *cloud computing*. Y por último, en la Sección 3.6 se validará la arquitectura presentada.

3.1. Requisitos

Para que un sistema distribuido de provisión de servicios multimedia satisfaga las necesidades más críticas debe verificar los siguientes requisitos:

Mínima latencia. Los flujos multimedia son especialmente sensibles a los retardos. Además por la propia naturaleza de la aplicación o el servicio proporcionado (especialmente los servicios que impliquen interactividad o consumo en tiempo-real) se exige que el retardo introducido sea mínimo.

Desacoplamiento. Los sistemas centralizados presentan una serie de problemas que son inherentes a su arquitectura. La provisión de servicios mediante una arquitectura centralizada puede implicar riesgos de robustez y escalabilidad. Por este motivo, la arquitectura propuesta debe desacoplar (en espacio, tiempo y a nivel de aplicación) a las entidades consumidoras de las productoras del servicio.

Heterogeneidad de capacidades y/o formatos. El sistema debe de permitir desplegar –a demanda o de forma automática– desplegar servicios de adaptación para así permitir la integración de generadores o consumidores de información heterogéneos.

Extensibilidad y adaptabilidad. Los requisitos de un sistema pueden variar a lo largo del tiempo. En el caso de los servicios multimedia puede ser debido a varios factores tales como añadir soporte a nuevos formatos de media, o añadir nuevas funcionalidades tales como nuevos operadores sobre flujos multimedia. Muchos de los sistemas existentes [22, 183, 202] no permiten añadir estas capacidades fácilmente. Por lo tanto, es deseable que la arquitectura propuesta permita añadir funcionalidades nuevas (transformación de resoluciones, mezcla de flujos, transcodificación, etc) de una forma sencilla y sin interrupción del servicio prestado.

Descubrimiento. Uno de los problemas que debe de afrontar un sistema multimedia es el de conocer los flujos multimedia disponibles, las características de los mismos y su localización (Uniform Resource Locator (URL)). El sistema propuesto debe permitir a los consumidores multimedia descubrir los flujos existentes así como las características y metadatos (*metadata*) de los mismos de forma automática.

Escalabilidad. Una de las dimensiones más importante en el diseño de un sistema distribuido es su escalabilidad. En el caso de un sistema multimedia este hecho se acentúa. Esto es debido a que algunos flujos, especialmente los de vídeo, requieren un ancho de banda mayor y su procesamiento es, por lo general, muy demandante en recursos. Por lo tanto, el sistema debe proporcionar los servicios sin merma en las prestaciones para un gran número de usuarios y/o flujos o al menos debe de habilitar mecanismos que permitan soportarlos.

Integración. Actualmente existe gran cantidad de dispositivos con capacidades multimedia (por ejemplo, cámaras IP, sensores, tabletas, teléfonos móviles, etc). Es deseable que el sistema propuesto utilice tecnologías que permitan la integración no disruptiva de cualquier dispositivo con independencia de sus particularidades.

Uso de estándares. El sistema desarrollado debe de hacer uso en la medida de lo posible de tecnologías que estén especificadas por organismos internacionales de estandarización. Con ello se evita el problema de la dependencia de proveedor (*vendor lock-in*). De este modo si el proveedor de una determinada tecnología empleada en el sistema deja de dar soporte a la misma, este podrá ser remplazado con un impacto mínimo para el sistema. De igual manera, adoptar tecnologías normalizadas es un importante paso para garantizar la interoperabilidad del sistema desarrollado con otros sistemas ajenos.

Seguridad. Los flujos multimedia pueden transportar información sensible que puede requerir un acceso controlado, además de exigir su integridad -es decir, proveerse con garantías de no ha sido modificada–. Un claro ejemplo de esta necesidad son

todos los servicios relacionados con vídeo-vigilancia. En general, la provisión de servicios multimedia debe garantizar la confidencialidad, la integridad y autenticación de las entidades involucradas.

Una vez identificados los requisitos que debe verificar el diseño de la arquitectura, a continuación se procederá a explicar los fundamentos de Extensible Multimedia Distribution Service (EMDS), una propuesta de arquitectura extensible para la provisión de servicios multimedia.

3.2. Diseño

Extensible Multimedia Distribution Service (EMDS) es la propuesta que se hace en el marco de esta Tesis para la provisión de servicios con contenido multimedia.

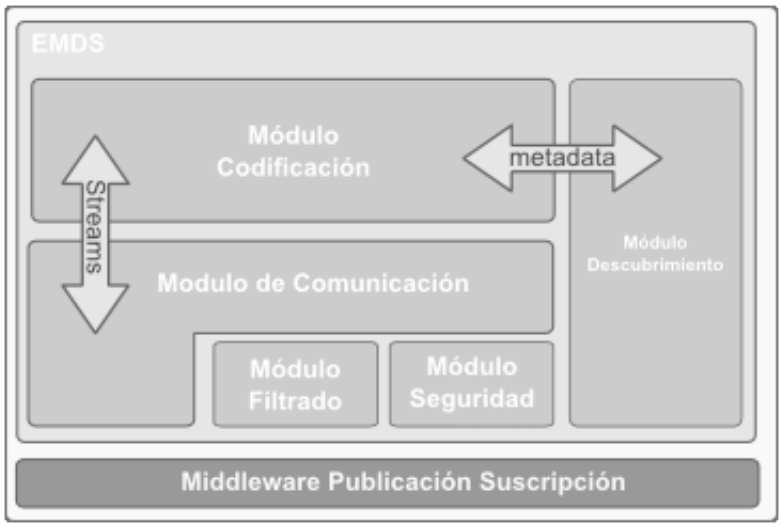


Figura 3.1: Módulos de EMDS

Antes de explicar los diferentes módulos constituyentes (ver Figura 3.1), se explicarán las decisiones generales adoptadas en el diseño de EMDS.

Uno de los pilares sobre los que se sustenta esta Tesis es la adopción de una aproximación publicación/subscripción centrada en datos. Los motivos para esta elección son los siguientes.

- En primer lugar, al adoptar el paradigma publicación/subscripción para el intercambio de datos, EMDS se beneficia de las ventajas y características –ya comentadas– de este modelo frente al paradigma cliente/servidor.

- Por otro lado, la faceta *data-centric* presenta una serie de ventajas frente a los otros esquemas de publicación/subscripción orientados a mensaje –MOM–. El modelo *data-centric* asocia a cada tópico publicado un tipo de dato que puede ser interpretado por el *middleware*. Esto implica que el *middleware* centrado en datos puede analizar el contenido de las muestras de los tópicos, lo que permite gestionar las muestras en base a su contenido. De esta manera se simplifican la realización de operaciones más complejas tales como el mantenimiento de un histórico, el almacenamiento temporal (en cache) de información o el filtrado de la misma.
- Por contra, en el modelo MOM los datos transportados son opacos al *middleware*. Esto es, los MOM al tratar el contenido de cada muestra como un conjunto de *bytes* sin un tipo asociado, dificultan la creación de servicios genéricos multimedia, ya que trasladan a la aplicación las operaciones relacionadas con el contenido de los mismos.

A continuación, para explicar el diseño propuesto se identifican cada uno de los módulos previstos así como las características más relevantes de cada uno de ellos.

3.2.1. Módulo de descubrimiento

Uno de los problemas a resolver en los sistemas distribuidos en general –y para la provisión de servicios multimedia en particular– es el descubrimiento de los de orígenes de datos disponibles. En el caso de multimedia, aparece un problema añadido: no sólo hay que identificar los publicadores disponibles en un determinado momento, sino que además hay que determinar las características de los flujos publicados (*metadatos*). Los *metadatos* asociados a un flujo en particular contienen la información que debe utilizar el subscriptor para interpretarlo correctamente. De esta manera, el descubrimiento desacopla las aplicaciones o servicios desplegados en un espacio de datos concreto de los flujos disponibles y de sus particularidades.

A continuación se listan algunos de los *metadatos* que deben incluirse en un sistema de descubrimiento:

Identificador (*id*). A cada flujo multimedia se le asigna un identificador de tipo Universally Unique IDentifier (UUID) [107] que permite identificarlo unívocamente de entre todos los presentes en el espacio de datos. Este identificador permite a los subscriptores seleccionar a qué flujos desean suscribirse.

Tipo de media (*media*). Este *metadato* especifica el tipo de señal asociada a un determinado flujo. Esto campo permite identificar si un determinado tópico transporta por ejemplo audio, vídeo, imágenes estáticas, texto, o incluso datos en bruto.

Codificación (*codec*). Un determinado contenido o *media* puede codificarse de múltiples maneras y formatos utilizando diferentes algoritmos de codificación (*codec*) y/o contenedores. Esta información la utiliza el subscriptor para seleccionar el decodificador (o el contenedor) adecuado y poder así procesar adecuadamente el flujo multimedia asociado.

Tasa de datos (*bitrate.*) Los flujos de multimedia tienen asociada una determinada razón o tasa de datos asociada al correspondiente *codec* utilizado. Esta información permite determinar cuáles son los requerimientos de ancho de banda mínimos necesarios para poder transmitir un flujo en determinado.

Palabras clave (*keywords*). Este campo permite asignar palabras clave a un flujo multimedia. Estas palabras clave permiten caracterizar de alguna manera ciertos flujos, posibilitando así descubrir aquellos que cumplan determinadas condiciones.

Codec data (*codec-data*). Algunos decodificadores requieren ser configurados previamente antes de poder empezar a descodificar información. La información de configuración normalmente se encuentra disponible en las cabeceras de los ficheros contenedores de los correspondientes flujos multimedia. Esta información debe de estar disponible en los decodificadores antes de empezar la codificación, por que debe de considerarse en la fase de descubrimiento. En este campo, por tanto almacenará toda la información necesaria para configurar los descodificadores.

Seguridad (*encryption* y *authentication*). En algunas ocasiones puede ser necesario proteger algunos flujos multimedia para evitar su acceso por terceros no autorizados, garantizar su integridad y/o autenticar a las entidades involucradas. En este campo se indicará si el flujo está protegido mediante algún mecanismo de seguridad.

Referencias (*original-id*). En el caso de flujos multimedia procesados o derivados de otros ya existentes, opcionalmente se puede incluir una referencia al flujo original o en el caso de flujos combinados, una lista de flujos contribuyentes a la mezcla. Este campo consta por tanto de una secuencia de cero o más elementos que apuntan a otros *id*.

Tiempo de referencia (*time-base*). Este campo representa la unidad de tiempo fundamental en la que se expresan las marcas de tiempo (*pts*) de los flujos (ver Sección 3.2.3).

Además de estos metadatos genéricos, existen otros que son específicos de cada determinado tipo de medio. Así por ejemplo en el caso de flujos visuales podemos encontrar los siguientes:

Resolución (*resolution*). En este campo permite especificar la resolución espacial de un determinado flujo de vídeo o imagen estática (*i.e.* el ancho y alto en píxeles).

Cuadros por segundo (*framerate*). Permite especificar la resolución temporal de un flujo de vídeo, esto es el número de cuadros por segundo.

Y en el caso de flujos de audio:

Frecuencia de muestreo (*samplerate*). Especifica el número de muestras por segundo (o frecuencia) con la que fue muestreada la señal de audio.

Número de canales (*channels*). Especifica el número de canales que codifica una flujo de audio. Así por ejemplo puede especificar si es monoaural (1 canal), estéreo (2 canales) o multicanal.

Además de estos metadatos definidos *a priori*, el módulo de descubrimiento permite al usuario o al servicio agregar sus propios metadatos. Con ello se puede hacer una adaptación a las necesidades concretas de un determinado escenario. Por ejemplo, en un escenario de vídeo-vigilancia, podrían incorporarse metadatos que se refieran a la localización de la cámara que captura un flujo de vídeo dado o por ejemplo un metadato podría añadir credenciales adicionales o identificar la Private Key Infrastructure (PKI) involucrada.

No obstante, para poder soportar la posibilidad de añadir metadatos y facilitar así su manejo y procesamiento, estos deben ser codificados de alguna manera que así lo facilite. En EMDS se ha elegido JSON como formato de codificación [39]. Los motivos para esta elección son los siguientes:

- JSON goza de gran popularidad [1].
- JSON es un lenguaje independientemente de lenguaje. Existen multitud de librerías escritas en multitud de lenguajes de programación que dan soporte a JSON, incluyendo Python, C y Java [95].
- Facilita su integración con servicios web [142].
- Es más ligero y compacto que otros lenguajes estructurados como XML [159].
- JSON es un formato *human readable*.

Proceso de descubrimiento. Los metadatos que han sido previamente definidos, deben estar disponibles por parte los subscriptores antes de empezar a recibir cualquier flujo multimedia. Por este motivo deben difundirse en el proceso de descubrimiento de flujos. EMDS realiza el descubrimiento de flujos mediante el descubrimiento estándar especificado en RTPS, el protocolo de transporte que define la familia de estándar DDS. Este protocolo, conocido como Simple Discovery

[1] http://visitmix.com/writings/the-rise-of-json

Protocol (SDP) permite el descubrimiento de entidades DDS. En esta Tesis se propone aprovechar el tráfico SDP para el descubrimiento de flujos EMDS, evitando así la generación de tráfico extra. Algunos autores han demostrado que es posible adaptar SDP en función de determinado requerimiento. Así por ejemplo en [162] se propone una extensión que permite aumentar la escalabilidad de SDP por medio de filtros de Bloom.

Listado 3.1: Metadata para flujo de vídeo

```
{
  "src-id": "b374bfb9-64d1-47b5-8329-f4d46c388e91",
  "media": "video",
  "codec": "H.264",
  "codec-data":""Ah4bAXZvcmJpcwAAAAACRKwAAAAAAACAtQEAAAAAALgBA3
     ..."
  "width": 1920,
  "height": 1080,
  "height": 1080,
  "bitrate": 4164385,
  "framerate": 23.976024,
  "encryption": false,
  "authentication": true,
  "origin-id": "eb1b146d-e2a3-4487-869b-597639c9ad43",
  "keywords": ["outdoor","camera","sector-1"],
  "time-base": "1/1000"
  }
```

Para poder llevar a cabo la inclusión de los metadatos necesarios en el proceso de descubrimiento de entidades EMDS sin romper la compatibilidad hacia atrás con otras aplicaciones DDS, se propone el uso de la política de calidad de servicio *USER_DATA* para el transporte de metadatos de flujos multimedia anteriormente descritos. La información almacenada en esta calidad de servicio, es almacenada en formato JSON por EMDS y diseminada por el SDP a todos los posibles suscriptores EMDS, que en base a esta información pueden decidir suscribirse a qué flujo suscribirse.

3.2.2. Módulo de codificación

La aproximación más sencillas a la hora de distribuir contenido multimedia consiste en publicar los flujos en bruto (*raw*), ya que en este caso el coste computacional de la compresión sería nulo. Sin embargo, esta aproximación no es generalmente válida ya que los flujos en bruto poseen una redundancia elevada especialmente si la calidad de los flujos es alta. Debido a esto, es deseable reducir la tasa de datos de los flujos por medio de algoritmos de codificación específicos que compriman

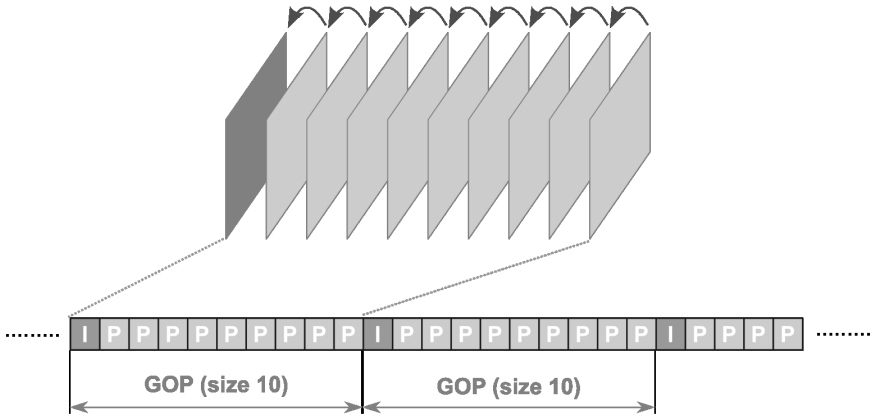


Figura 3.2: Codificación de la señal de vídeo: GOP.

(reduzcan la redundancia) sin afectar de una forma significativa a la calidad final percibida.

El módulo de codificación se encarga de codificar y decodificar los flujos multimedia que son transportados usando EMDS. EMDS incluye los codificadores más populares y que implican un buen compromiso entre calidad resultante y bit-rate para los distintos tipos de media considerados (en concreto *H.264* [181] y *VP8* [11] para vídeo, *Vorbis* [120] para audio, entre otros). Además el diseño de este módulo permite añadir soporte a nuevos *codecs* de una manera sencilla. Para ello se proporcionan interfaces que abstraen las operaciones comunes a todos los codificadores/-decodificadores de flujos multimedia, tales como inicialización, codificación y decodificación. De este modo, el usuario únicamente debe debe de implementar dichas operaciones para dar soporte a nuevos formatos y así quedar integrado dentro de EMDS. De este modo se consigue que el diseño de EMDS no se encuentre vinculado a un formato de codificación en concreto como ocurre en otros sistemas [45].

Uno de los mayores problemas que tiene que afrontar este módulo es el sincronismo entre productores y consumidores de flujos. La aproximación publicación/-suscripción hace que publicadores multimedia y los correspondientes subscriptores se encuentren desacoplados temporalmente, lo que puede dificultar la sincronización. Esto es debido a que los publicadores no conocen qué suscriptores se encuentran suscritos al flujo que producen, y por lo tanto es difícil establecer puntos de sincronización desde donde comenzar un proceso de decodificación. Aunque esto no supone un gran problema para algunos *codecs* que establecen puntos de sincronización periódicamente, para otros –especialmente algunos de vídeo explicados más adelante– debe ser tenido en cuenta.

El ejemplo más evidente de este hecho ocurre en los flujos de vídeo. Por lo ge-

neral, dado la elevada redundancia temporal que exhibe la señal de vídeo (y otras), para obtener un mejor compromiso entre el *rate* generado y la distorsión introducida se usan codificadores diferenciales. Estos, en lugar de codificar la señal (o cuadro actual) codifican la diferencia o predicción realizada a partir de un instante de tiempo (o cuadro) anterior (puede que compensado en movimiento).

Para evitar que un error en la recepción de un cuadro se acumule y genere errores en el flujo decodificado indefinidamente, la codificación se realiza segmentando la señal en los así denominados GOP, tal que el codificador supone que entre dos GOP no existen ningún tipo de dependencia. Un GOP se compone de una primera imagen (o cuadro) codificada sin dependencias (cuadro I, *Intra-frame* también denominado *keyframe*) y una serie de imágenes (cuadros) que toman uno o varios cuadros precedentes para realizar una predicción (cuadros P, *Predictive*). Algunos codificadores, consideran una variante conocida como *Bidirectional* o cuadro P, que utiliza además cuadros sucesivos para estimar las predicciones. Por tanto, un *GOP* es el conjunto de cuadros (P o B) entre dos cuadros de tipo I.

El carácter asíncrono de las subscripciones tiene dos implicaciones. Puede ocurrir que el primer GOP no se reciba completo, es decir no podrá ser decodificado por no disponer del correspondiente cuadro I, lo que introducirá un retraso inicial en la reproducción. Este retraso se puede reducir si se reduce el tamaño máximo de los GOPs. Ahora bien, esto implicará un mayor *bit-rate*. Para minimizar los retardos, EMDS toma por defecto un tamaño de GOP igual al número de cuadros correspondientes a 1 segundo [2], es decir a *framerate* cuadros, de modo que en el peor caso transcurrirá un segundo desde que publicador y suscriptor se descubren hasta que comienza la decodificación. No obstante, en función de los requerimientos podrá aumentarse o reducirse el tamaño del GOP a costa de aumentar la latencia de inicio o la resistencia ante errores respectivamente.

3.2.3. Módulo de comunicación

El componente más importante de un sistema multimedia distribuido es el encargado de comunicar a los distintos componentes entre sí. EMDS usa el paradigma de publicación/suscripción centrada en datos para este menester. Como se ha mencionado anteriormente, EMDS se construye sobre la especificación del *middleware* DDS.

Este módulo es el encargado de distribuir y recibir los paquetes correspondientes a los flujos multimedia en el sistema. Concretamente, las funciones de este módulo son:

[2] http://www.avidemux.org/admWiki/doku.php?id=tutorial:h.264

- En el publicador, se encarga de transformar los paquetes provenientes del codificador en tópicos DDS.
- En el suscriptor, su función es la de transformar las muestras de tópicos provenientes del *middleware* DDS en paquetes que serán procesados por los decodificadores adecuados.

Como se ha mencionado anteriormente, una de las principales ventajas de usar una aproximación *data-centric* (no opaca a los datos) frente a una aproximación *message-centric* es que el *middleware* tiene acceso al tipo de dato que transporta cada mensaje, de modo que puede acceder incluso al contenido de los mismos permitiendo así al *middleware* diversos tipos de operaciones tales como el almacenamiento temporal en cache de la información y su filtrado.

Debido a este hecho, cobra vital importancia el diseño del tipo de dato de los tópicos DDS empleados para transportar los flujos multimedia, ya que esto será determinante para sacar el mayor partido de la aproximación *data-centric*. Así por ejemplo, utilizar un tópico que únicamente se limite a incluir los paquetes generados por el codificador de vídeo, no aprovecharía las posibles ventajas de la aproximación *data-centric*, ya que pueden usarse campos del tópico de flujo para incluir información extra que permita una mejor experiencia, tales como etiquetado y filtrado.

A continuación se describe la estructura de tópico que EMDS propone (ver Listado 3.2):

stream-id. Este campo permite identificar unívocamente los flujos multimedia existentes en el espacio de datos DDS mediante un identificador de tipo Universally Unique IDentifier (UUID). Gracias a la aproximación *data-centric* es posible por tanto suscribirse únicamente a los flujos con un determinado identificador.

priority. Este campo permite asignar una prioridad a cada paquete que conforma el flujo multimedia. Así por ejemplo, este módulo asigna la máxima prioridad a los paquetes esenciales para la decodificación del flujo tales como parámetros de configuración del decodificador o los *keyframes* y asigna prioridades menores a paquetes con información que sólo mejora la calidad del *stream* pero que no son esenciales para la decodificación del mismo. Este mecanismo permite por ejemplo implementar flujos de vídeo multicapa de una manera sencilla mediante el filtrado de paquetes a partir de una determinada prioridad [110].

payload. Este campo contiene los datos que deben de pasarse al decodificador, es decir información multimedia codificada o parámetros para el mismo.

flags. En este campo se especifican algunos *flags* que permiten identificar el tipo de paquete que se transporta en este campo y algunas características del mismo. Por

Listado 3.2: Estructura de los tópicos EMDS.

```
typedef octet[40] UUID;

struct Dictionary{
  string name;
  string value;
};

typedef octet[20] Signature;

struct MediaStream{
  UUID                  id; //@key
  short                 priority;
  sequence<octet>       payload;
  unsigned long         flags;
  Dictionary            tags;
  unsigned long long    pts;
  unsigned long long     sequence_number;
  Signature             authentication;
};
```

ejemplo se puede especificar si el paquete contiene datos para decodificar un flujo, si incluye parámetros del codificador, si el campo *payload* se encuentra cifrado o incluso la notificación de algunos eventos, como por ejemplo el fin de un flujo (End Of Stream (EOS)).

tags. Este campo consiste en una serie de cadenas de texto que permiten añadir información textual al flujo multimedia. Esto permite al usuario añadir información que puede ser usada para enriquecer el contenido del flujo. Ejemplos de uso puede ser indicar la existencia de objetos móviles en un flujo de vídeo, coordenadas GPS de una cámara móvil, etc.

pts. Este campo incluye una marca de tiempo tomando como referencia a un reloj maestro. Esta marca de tiempo permite la sincronización de flujos relacionados (*p.e.* sincronizar audio y vídeo en un reproductor).

sequence_number Este número de secuencia identifica el número de cuadro que representa.

authentication. Este campo incluye una firma digital del contenido enviado en el paquete. Esto permite verificar el contenido del mismo, dificultando los ataques contra su integridad por parte de usuarios maliciosos.

Un hecho relevante respecto a este módulo es que algunos campos son opcionales. Así por ejemplo, es opcional que un paquete incluya una autenticación o etiquetas (*tags*). Esto se implementa gracias al empleo del estándar de tipos extensibles X-Types de DDS [135], que permite la omisión de algunos campos cuando no son necesarios, reduciendo así el tamaño de mensaje y el ancho de banda. Esto además permite que futuras versiones de EMDS puedan incluir nuevos campos sin perder la compatibilidad con versiones anteriores, facilitando así el mantenimiento y evolución de los sistemas EMDS.

Otro hecho destacable es que como se puede observar los campos descritos anteriormente no incluyen metadatos acerca del flujo, únicamente incluye información acerca del paquete actual. Esto es debido, a que como se especificó en la Sección 3.2.1, dichos metadatos son enviados a los suscriptores en la fase de descubrimiento.

3.2.4. Módulo de filtrado

Una de las principales ventajas de adoptar una aproximación *data-centric* es la posibilidad de realizar operaciones avanzadas sobre los datos en función de su contenido. Más concretamente, EMDS permite filtrar flujos multimedia, de modo que únicamente sean visibles por parte de los subscriptores algunos flujos que cumplan

determinadas condiciones. En EMDS es posible realizar filtrado de subscripciones a varios niveles:

Palabras clave. Durante el proceso de descubrimiento de flujos multimedia, los publicadores pueden especificar dentro de los metadatos palabras clave (*keywords*). Estas palabras clave permiten a los suscriptores identificar más específicamente a qué flujos desean subscribirse. Un ejemplo de esta funcionalidad en un escenario de vídeo-vigilancia podría ser seleccionar las cámaras de vigilancia situadas en el exterior (es decir, aquellas publicaciones que contengan el término *outdoor* entre las palabras clave).

Prioridades. Dentro de algunos flujo multimedia no todos los paquetes tienen la misma relevancia: mientras algunos son esenciales para la decodificación del flujo, otros no impiden la decodificación del flujo si no se tienen. Por ejemplo en un flujo de vídeo codificado con *VP8*, mientras que la pérdida del cuadro de referencia (cuadro I) o los parámetros de configuración significa una decodificación incorrecta de todo el GOP, los daños ocasionados por la pérdida de un cuadro tipo P son mucho menores. Debido a esto se puede considerar que en situaciones de inestabilidad de la red, es preferible priorizar la recepción de paquetes esenciales para la decodificación en el destino. Este mecanismo puede ser empleado para habilitar mecanismos de escalabilidad temporal en escenarios donde el ancho de banda sea reducido (por ejemplo, eliminando todos los cuadros de tipo P).

Etiquetado. Una última manera de realizar filtrado en EMDS es mediante etiquetas incrustadas en el propio flujo. Así por ejemplo, se pueden añadir etiquetas a determinados paquetes del flujo a modo de atributos diferenciadores. Esto permite suscribirse únicamente a los fragmentos de un flujo que hayan sido etiquetados. Un posible uso de este filtrado en el escenario de vídeo-vigilancia permitiría por ejemplo la subscripción solamente a los fragmentos de vídeo con una etiqueta tal que corresponda a objetos en movimiento.

3.2.5. Módulo de seguridad

Uno de los aspectos novedosos de EMDS es la inclusión de mecanismos de seguridad. La motivación de esta necesidad reside en algunos escenarios en los que la información multimedia transmitida es sensible y debe ser protegida ante posibles accesos o alteraciones no autorizados.

El estándar DDS permite usar diferentes protocolos de transporte –incluyendo algunos seguros como Datagram Transport Layer Security (DTLS) [156] o Transport Layer Security (TLS) [47]–.

No obstante, para mejorar la seguridad el *middleware* debe incluir en su especificación de forma explícita cómo proporcionar un servicio seguro (en todas sus facetas). Aunque, el RFC sobre DDS Security ha sido aprobado [134] en el momento de elaboración de la Tesis todavía no se ha seleccionado cuál será la especificación finalmente estandarizada.

Por estos motivos, se ha considerado que el diseño de EMDS debe contemplar los siguientes aspectos básicos de seguridad:

Autenticación. Este aspecto garantiza la identificación fehaciente de las entidades involucradas. La provisión de este aspecto es importante ya que de otra forma la veracidad de un determinado flujo no puede garantizarse.

Integridad. El objetivo es garantizar la detección de posibles alteraciones (voluntarias o involuntarias) del contenido de los flujos multimedia. En determinados entornos, es de vital importancia poder asegurar que los flujos recibidos corresponden con los flujos que fueron publicados originalmente. Un entorno donde se evidencia este requisito es en vídeo-vigilancia: si este aspecto no estuviera garantizado un atacante podría desplegar un servicio consistente en publicar un flujo de vídeo alterado.

Privacidad. Esta aspecto tiene por objetivo evitar el acceso no autorizado al contenido multimedia.

Para proporcionar los dos primeros aspectos, EMDS permite firmar el contenido de cada mensaje perteneciente a un flujo mediante un código Hash-based Message Authentication Code (HMAC) [104]. HMAC es un código de comprobación que se genera mediante una función *Hash* a partir del contenido de los campos que se desean asegurar y una clave secreta compartida por las entidades emisora y receptora.

EMDS incluye los siguientes campos (ver Sección 3.2.3) en el cálculo del código HMAC: identificación (*stream-id*), carga (*payload*) y tiempo de presentación (*pts*). El código HMAC resultante es almacenado en el campo *authentication*.

EMDS dispone de dos modos para el cifrado de los flujos multimedia: parcial o total. En el cifrado total, todos y cada uno de los paquetes correspondientes al flujo se cifran mediante una clave compartida. Sin embargo, en escenarios donde es necesario minimizar los retardos y la carga computacional, se puede hacer uso del cifrado parcial. En este modo, se cifran únicamente los paquetes que son esenciales para la decodificación del flujo, tales como los *keyframes* en el caso de vídeo o parámetros de configuración del decodificador. En algunos casos –para vídeo codificado con H.264 o VP8 dependerá del tamaño medio del GOP– esto puede suponer un ahorro significativo en el número de operaciones de cifrado realizadas: por

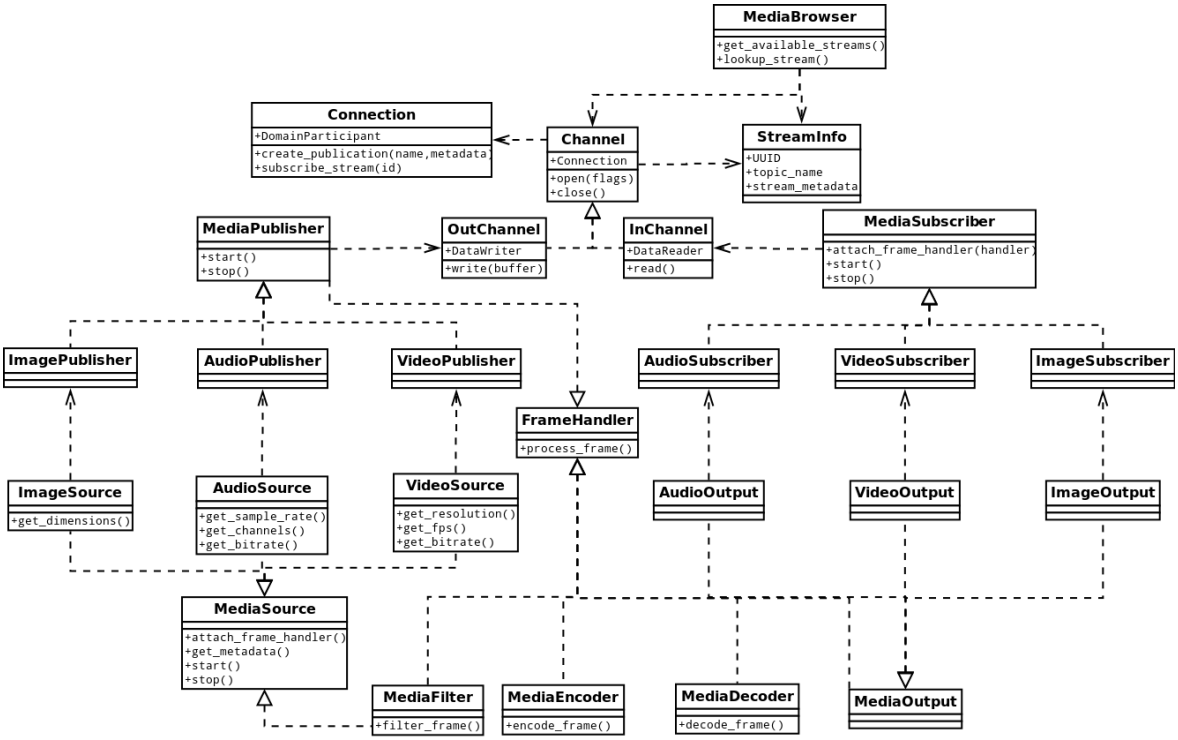


Figura 3.3: Jerarquía de clases de EMDS.

ejemplo en un flujo de vídeo con tamaño de GOP igual a 30, solo se realizaría una operación de cifrado cada 30 cuadros. El cifrado parcial de información, no solo es aplicable a flujos de vídeo, sino que es aplicable a todos los flujos que utilicen diferenciación diferencial, ya que si ciframos las muestras que sirven de referencia para la codificación de muestras sucesivas la señal original no podría reconstruirse si no se aplica el proceso de descifrado (*e.g.* codificación *ADPCM* en audio).

3.2.6. Diagrama de clases

La arquitectura de EMDS se ha diseñado para permitir la reusabilidad y modularidad del código. En la Figura 3.3 se muestra la jerarquía de las clases más relevantes de EMDS. Es destacable la sencillez de la jerarquía y de la API asociada.

A continuación se describen los componentes más importantes de EMDS (Figura 3.3):

MediaSource. Los *MediaSource* y sus clases derivadas son los puntos de entrada de un flujo multimedia dado en EMDS: leen un flujo multimedia y lo publican dentro de EMDS. El origen del flujo puede ser un dispositivo hardware, un flujo RTP o un fichero.

MediaOutput. Los *MediaOutput* y sus clases derivadas son la salida hacia el exterior de los flujos gestionados por EMDS. Al igual que los *MediaSource* los destinos de los flujos multimedia pueden ser dispositivos hardware, flujos RTP o ficheros.

MediaEncoder/MediaDecoder. Son los componentes encargados de codificar/decodificar los flujos multimedia utilizando un determinado *codec*.

Connection. Este componente representa la asociación de un componente EMDS con el *middleware* empleado. Este componente actúa como una fachada ante la *API* del *middleware*, de modo que el resto de componentes EMDS puedan usar diversos *middleware* de manera transparente.

Channel. Este componente representa un canal virtual que se establece con el *middleware* para el intercambio de datos. Este canal puede ser de entrada (suscripción) o de salida (publicación). El cometido de este componente es el de realizar las operaciones de lectura escritura en una conexión (*Connection*). Un *Channel* está asociado con un determinado tópico DDS.

MediaPublisher. Este componente es el encargado de publicar un flujo multimedia dentro de EMDS por medio de un *Channel*.

MediaSubscriber. Este componente es el encargado de suscribirse a flujos multimedia presentes en EMDS.

MediaBrowser. Este componente es uno de los más importantes del sistema ya que permite descubrir y filtrar los flujos multimedia que están siendo publicados dentro del espacio de datos de *EMDS*. Gracias a este componente los *MediaSubscriber* pueden seleccionar a qué flujo suscribirse.

MediaFilter. Un *MediaFilter* es una interfaz que representa a un componente generador de un flujo multimedia a partir de otro por medio de alguna transformación o procesamiento. Es el componente esencial dentro de EMDS para el procesamiento distribuido de flujos. El usuario es el encargado de implementar *MediaFilters* de acuerdo a sus necesidades. Para ello basta con implementar los métodos que proporciona la interfaz, por ejemplo el método *filter*. Así por ejemplo un filtro de escalado de vídeo, debería implementar en el método *filter* el escalado de un cuadro de vídeo mediante interpolación u otro método. La salida de este filtro sería un nuevo cuadro de vídeo con una nueva resolución espacial.

FrameHandler. Un FrameHandler es una clase abstracta que representa la interfaz que deben de implementar todos los componentes EMDS que reciban un marco desde otro componente. Por ejemplo, un codificador (MediaEncoder) desea recibir los frames originados en un (MediaSource) para codificarlos, por lo tanto debe implementar esta interfaz.

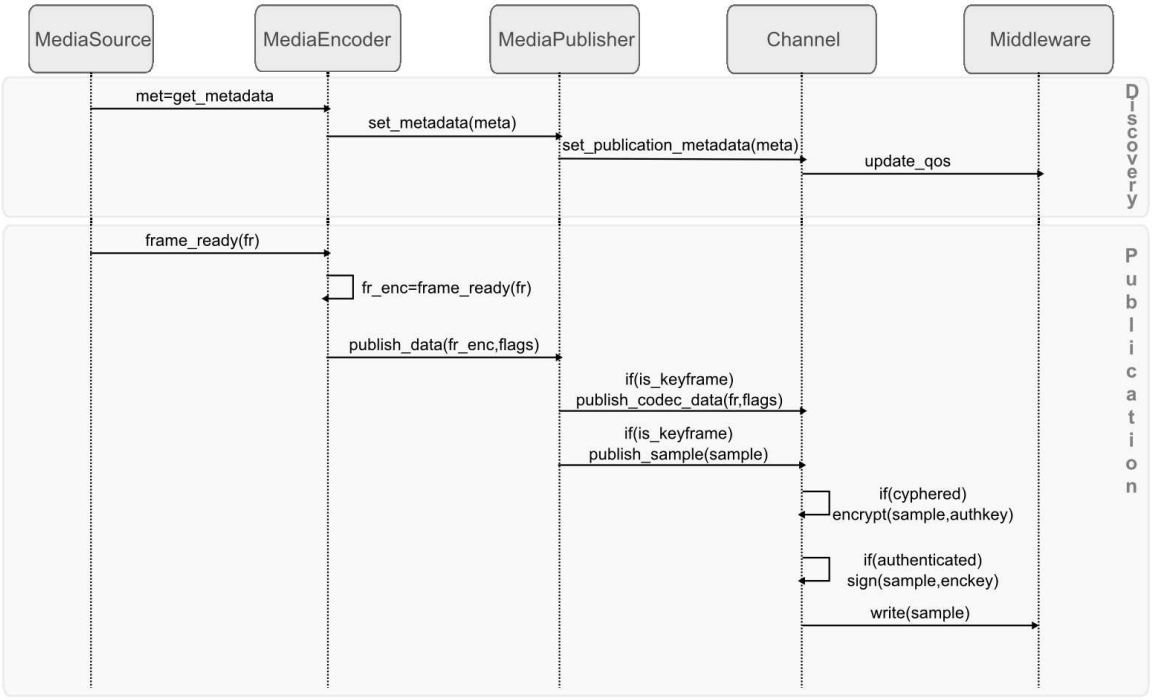


Figura 3.4: Diagrama de secuencia: Publicación de un flujo de video

En la Figura 3.4 se muestra un ejemplo de cómo interaccionan algunos de los componentes de EMDS descritos anteriormente . Concretamente se observa la secuencia de operaciones e interacciones necesarias para la publicación de un flujo multimedia. Esta secuencia de operaciones es independiente del tipo de media y codificación aplicada. En la parte de arriba se observan las operaciones necesarias para iniciar el proceso de anuncio de un flujo multimedia, donde se especifican los metadatos que corresponden al flujo. En la parte inferior de la figura, se observa el proceso para la publicación de las muestras correspondientes a un determinado flujo.

3.3. Implementación

Tras la descripción del diseño de EMDS (bloques funcionales y diagrama de clases), en esta Sección se proporcionan brevemente algunos detalles de la implementación de EMDS realizada, la cual será evaluada posteriormente en la Sección 3.6.

Lenguaje utilizado El núcleo de EMDS ha sido implementado haciendo uso de ANSI C para favorecer la eficiencia en el manejo y procesamiento de flujos multime-

dia [148]. Otro motivo para esta elección, es que siendo una arquitectura orientada al procesamiento multimedia, debe de facilitar la integración con librerías orientadas a este fin: la mayoría de librerías de tratamiento y codificación multimedia están implementadas en C/C++ o le dan soporte [70, 112, 136].

Elección de middleware De entre los posibles *middleware* normalizados (ver Capítulo 2), en esta Tesis se ha elegido el estándar DDS como núcleo de comunicación del sistema. Las ventajas de esta elección son múltiples: DDS es la única especificación estandarizada existente para sistemas de publicación/subscripción centrados en datos. Concretamente, DDS especifica tanto la API [132] como el protocolo de comunicación [133], pudiendo elegir así entre múltiples implementaciones, y finalmente DDS es el estándar de publicación/subscripción que incluye la mayor cantidad de políticas de calidad de servicio (QoS).

Aislamiento de subsistemas Una manera de mejorar la escalabilidad del sistema es la agrupación de publicadores y suscriptores de medios según intereses comunes. Por ejemplo, pueden existir varios subsistemas EMDS no relacionados entre sí funcionando en el mismo entorno de red, por lo que no queremos que un subsistema consuma los flujos del otro y viceversa. Para llevar a cabo esto, EMDS propone utilizar la calidad de servicio `PARTITION` de DDS. Gracias a esta calidad de servicio, se puede particionar el espacio de datos global en grupos aislados que no comparten información. Así, se pueden configurar los publicadores y subscriptores EMDS para que pertenezcan a una o varias particiones.

Soporte de codecs El abanico de codecs multimedia disponibles es inmenso. En EMDS se incluyen por defecto algunos de los más populares, tales como *H.264* en el caso de vídeo o *Vorbis* en el caso de audio. No obstante, el diseño propuesto por EMDS permite añadir fácilmente soporte para nuevos codecs.

Filtrado de flujos En la sección de diseño, proponíamos el filtrado de determinados flujos para así que un suscriptor pueda recibir fragmentos del mismo que cumplan determinadas condiciones tales como tener alguna etiqueta o prioridad concreta. Para habilitar esta funcionalidad, se ha decidido utilizar *ContentFilteredTopics*. Mediante este tipo de tópicos, los suscriptores pueden especificar determinadas condiciones que deben de cumplir las muestras en base a su contenido. En el caso de EMDS este filtrado se realiza en base al contenido de los campos *priority* y *tags*.

Señalización En el caso de los flujos multimedia, es necesario incluir mecanismos de señalización para avisar a los suscriptores de algún cambio de relevancia. Así por ejemplo puede ocurrir, que un publicador cambie los parámetros de configuración de un codec de vídeo por algún motivo, o que bien necesite incluir puntos de sincronización para los suscriptores recién incorporados. Para ello, al comienzo de cada *GOP*, se publicarán –en caso de que el codec considerado soporte un cambio de sus parámetros configuración durante su ciclo de vida– los parámetros de codificación del codec, de modo que si un suscriptor comienza la suscripción en dicho punto, pueda configurar el decodificador en caso de ser necesario. Para ello, antes del primer cuadro de un GOP se intercala dicha información en el flujo multimedia, usando por tanto el mismo tópico, y permitiendo así que todos los suscriptores de ese flujo reconfiguren el decodificador. Así por ejemplo en el caso de flujos H.264, antes del comienzo de cada GOP se añaden los parámetros de configuración del flujo (Sequence Parameter Set (SPS) y Picture Parameter Set (PPS) de H.264 [89]).

Seguridad Para garantizar la privacidad, EMDS opcionalmente puede cifrar el contenido del campo *payload* evitando así la posible decodificación del flujo por parte de usuarios no autorizados. EMDS cifra el contenido del campo *payload* usando el algoritmo AES de clave simétrica [171]. Respecto a la autenticación, como se mencionó en la sección de diseño, EMDS utiliza HMAC. Concretamente se ha elegido la variante HMAC-SHA[48], que utiliza SHA [50] como función *Hash*. Se han descartado otras variantes tales como HMAC-MD5, por presentar problemas de seguridad [179].

3.4. Construcción de servicios EMDS

EMDS proporciona un conjunto de componentes que permiten de una forma sencilla la construcción y provisión de servicios multimedia de diversa índole. Dichos servicios permiten satisfacer de forma flexible diferentes necesidades –incluso cambiantes en el tiempo- para así enriquecer la experiencia del usuario final, objetivo último de todo sistema multimedia.

Este tipo de servicios son de gran utilidad en multitud de entornos, tales como el ya comentado de vídeo-vigilancia, los procesos de manufactura o realidad aumentada, entre otros.

La mayoría de los servicios que pueden implementarse en EMDS están concebidos siguiendo una aproximación *prosumer*, nombre que viene por la contracción de los términos *producer* and *consumer*. Esto significa que los servicios EMDS son a la vez consumidores (se subscriben a uno o varios flujos EMDS) y productores (generan nuevos flujos modificados/procesados/mezclados a partir de los originales).

Debido al desacoplamiento existente entre publicadores y subscriptores, estos servicios pueden desplegarse a demanda en cualquier momento en el escenario –estando el sistema en operación sin necesidad de interrumpir otros flujos o servicios– . Tras lo cual, una vez en ejecución y sin interrupción alguna, los nuevos *prosumers* procederán a realizar su tarea de manera automática, descubriendo y generando nuevos flujos que pueden ser nuevamente consumidos por otros servicios.

De este modo EMDS permite la construcción y provisión de servicios multimedia complejos de manera sencilla y escalable, funcionalidad esta que sería muy costosa proporcionar (sin fisuras ni interrupciones) por medio de una aproximación cliente/servidor.

Otra ventaja que proporciona el desacoplamiento aludido entre productores y consumidores de información es que al no estar ligados a una dirección física, los *prosumers* pueden desplegarse en cualquier ubicación de la red y si fuera necesario pueden ser migrados o replicados a otro *host* de manera totalmente transparente, lo que convierte a EMDS en una tecnología escalable apropiada para su ejecución en entornos de *cloud computing* (ver Sección 3.5)

En los siguientes sub-apartados dentro de esta Sección, se describen algunos de los servicios que pueden construirse gracias a EMDS.

3.4.1. Transformaciones

El primer tipo de servicios que se pueden construir son las transformaciones (Figura 3.5). Estas permiten generar nuevos flujos multimedia perceptualmente equivalentes al original pero con alguna modificación en algún parámetro (o metadato) del flujo codificado . El principal objetivo de este tipo de servicios es el de dar soporte a nuevos dispositivos (con capacidades heterogéneas) que *a priori* sean incompatibles con los flujos existentes.

Algunos ejemplos de servicios de transformación que pueden ser desarrollados con EMDS:

Reescalado. Cambio en la resolución del flujo multimedia. En el caso del vídeo corresponde a un cambio en la resolución espacial (altura y anchura en píxeles). Por otro lado, en el caso del audio, un ejemplo podría ser la modificación de la frecuencia de muestreo del flujo.

Transcodificación. Este tipo de servicios genera una réplica del flujo de entrada pero re-codificado con un *codec* diferente al original. Un ejemplo podría ser transformar un flujo de vídeo H.264 a VP8.

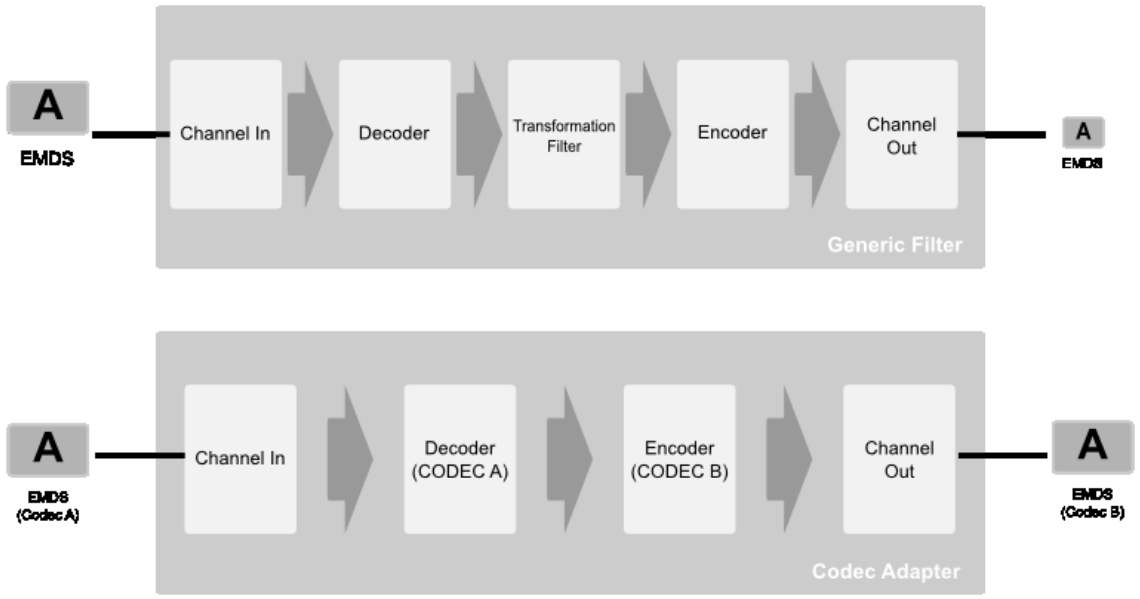


Figura 3.5: Servicio EMDS: Transformación de flujos.

Reducción de tasas de datos. Este tipo de servicios generan una copia del flujo de entrada generando un nuevo flujo con menor consumo de ancho de banda, con la potencial penalización de reducir la calidad del flujo resultante. Esto permite generar flujos a baja resolución para dar soporte a dispositivos con capacidades reducidas o accesibles a través de redes en desventaja.

Aunque se han presentado como servicios distintos, estos servicios se pueden combinar para realizar transformaciones híbridas: así por ejemplo se puede implementar un servicio que realice un escalado y posterior transcodificación del flujo.

3.4.2. Análisis

Este tipo de servicios obtienen información a partir de un análisis del contenido de los flujos multimedia involucrados. Para ello se suscriben a flujos multimedia y generan datos u estadísticas en base al análisis realizado. Dichas estadísticas pueden ser almacenadas o publicadas de nuevo por parte del usuario (Figura 3.6.

Un ejemplo de este tipo de servicios puede ser la identificación de locutores en un flujo de audio, el disparo de alarmas o por ejemplo la detección de movimiento u objetos en un flujo de vídeo. Las acciones a realizar con los resultados del análisis es elección del desarrollador del servicio (*e.g.* almacenar en un fichero, publicar como tópico DDS, invocar a un actuador, generar una llamada, etc).

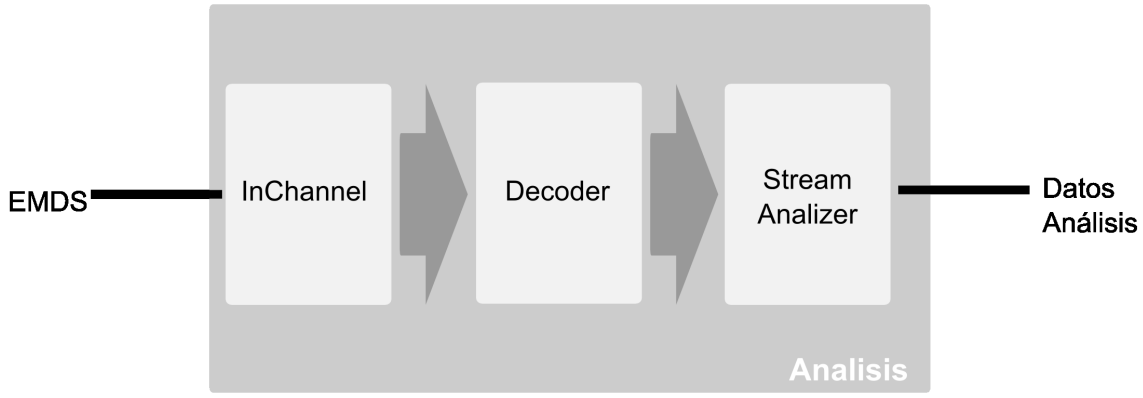


Figura 3.6: Servicio EMDS: Análisis de flujos

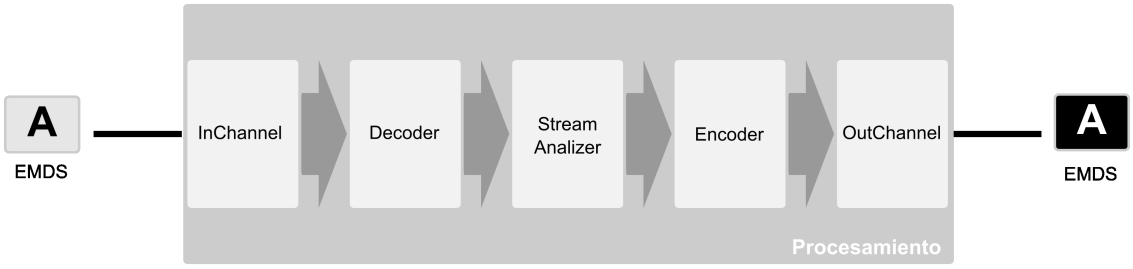


Figura 3.7: Servicio EMDS: Procesamiento de flujos.

3.4.3. Procesamiento

Este tipo de servicios a partir de un flujo de entrada generan un nuevo flujo resultado de algún algoritmo de procesamiento. Este tipo de servicios está relacionado con los servicios de transformación pero a diferencia de estos, en este caso el nuevo flujo es perceptualmente diferente al original.

Un campo que puede beneficiarse de este tipo de servicios es el de la visión artificial. Normalmente los algoritmos de visión artificial trabajan sobre flujos de vídeo donde cada fotograma es resultado de aplicar una operación de filtrado. Algunos ejemplos de este tipo de operadores son detección de fronteras, morfología matemática, etc.

Las ventajas que EMDS añade para los servicios de procesamiento son:

- Pueden desplegarse múltiples instancias de *prosumers* de procesamiento en paralelo, cada uno de ellos con unos parámetros de entrada distintos, obteniéndose así en paralelo distintos flujos.

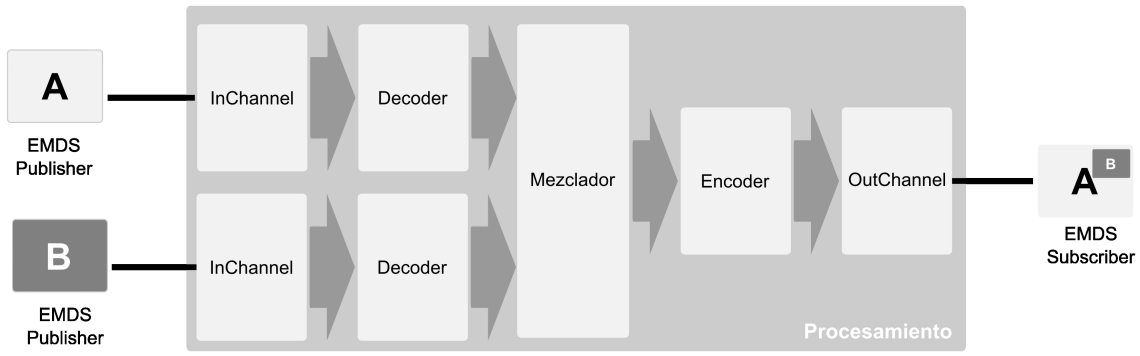


Figura 3.8: Servicio EMDS: Composición de flujos.

- El resultado de un servicio de procesamiento puede servir a la vez de entrada para otros *prosumers*, permitiendo así poder implementar procesamientos complicados de una forma sencilla.

3.4.4. Composición

Otro servicio de utilidad dentro de los servicios que se puede generar usando componentes *EMDS* es la composición de flujos multimedia. Este servicio consiste en la generación de un nuevo flujo a partir de la combinación de dos o varios flujos originales. La utilidad de esto es reducir el número de flujos a los que suscribirse aunque se necesite información de múltiples flujos. De este modo se reducen los requerimientos de ancho de banda y recursos necesarios en el suscriptor.

Ejemplos de aplicación de este servicio pueden ser:

- Mezclar de audio, para generar un flujo resultante que consuma el mismo ancho de banda que uno de ellos.
- Picture in Picture (PIP): Mostrar un flujo de vídeo a pantalla completa mientras otro flujo aparece representado en una pantalla menor.
- Generación de índices: Consiste en mostrar múltiples flujos en una cuadrícula para tener una visión global de los flujos publicados en el sistema.

3.4.5. Pasarelas

Hasta ahora los servicios presentados consumen flujos EMDS y producen nuevos flujos EMDS. Sin embargo en algunos escenarios es necesario importar y exportar

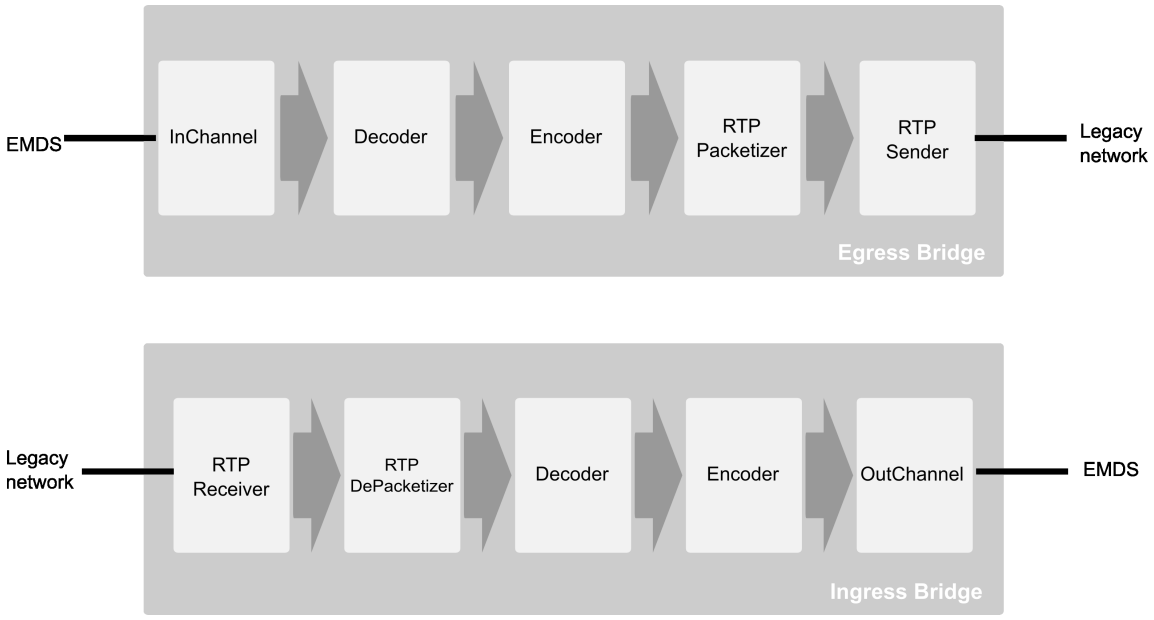


Figura 3.9: Servicio EMDS: Pasarelas

flujos multimedia EMDS desde y hacia sistemas que no soportan EMDS. El cometido de este servicio es por tanto permitir la integración de dispositivos sin soporte EMDS con un despliegue EMDS.

El ejemplo más claro de este tipo de servicios es el de ser una pasarela entre un productor/consumidor de vídeo basado en flujos RTP con uno basado en EMDS.

3.4.6. Realidad aumentada

Este tipo de servicios generan flujos multimedia que contienen información visual virtual sobreimpresa a la información física real. Para ello, este tipo de servicios toman información proveniente de un flujo EMDS y lo combinan con información proveniente de algún sensor o fuente de datos interna o externa al espacio de datos de DDS, como los provenientes de un sensor de sensor de posición, un acelerómetro, un servicio web o un servicio de análisis EMDS.

La principal utilidad es la de generar flujos de vídeo con información virtual incrustada para dispositivos que no disponen de dicha cualidad o tienen reducidas capacidades de procesamiento. Un ejemplo de esta funcionalidad es la sobreimpresión de información contextual en eventos deportivos: (*e.g.* imprimir el número, nombre o estadísticas sobre cada jugador en un partido de fútbol, o mostrar la telemetría proveniente de los sensores de un coche de fórmula 1.

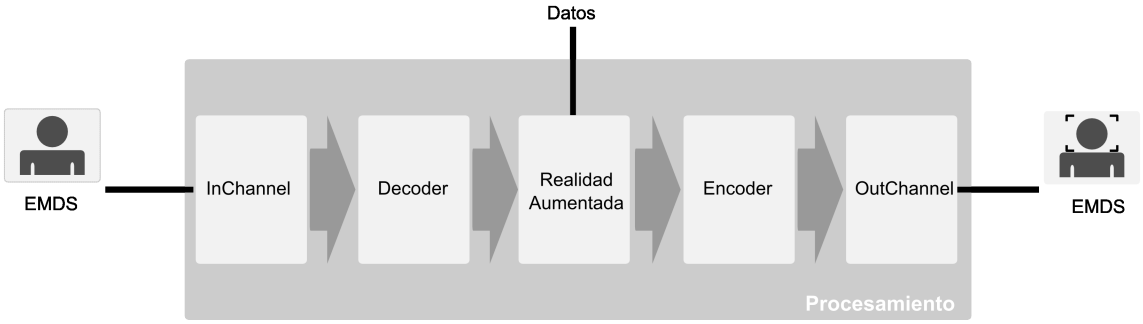


Figura 3.10: Servicio EMDS: Realidad aumentada.

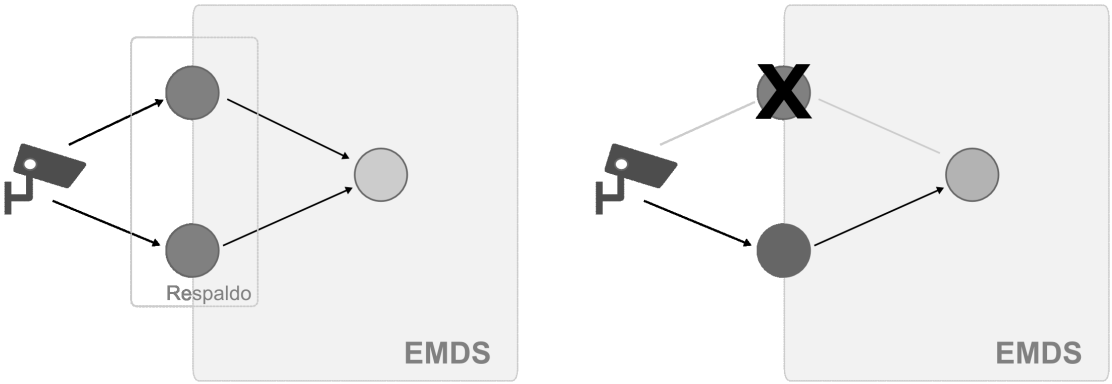


Figura 3.11: Servicio EMDS: Respaldo de flujos.

3.4.7. Respaldo

Finalmente, otro de las servicios que EMDS puede proporcionar viene gracias a la aproximación *data-centric*. Se trata de la generación de *backups* o respaldos de flujos multimedia de una manera sencilla para aumentar la robustez del sistema. Al existir un desacoplamiento espacial entre publicadores y suscriptores, múltiples publicadores pueden publicar copias del mismo flujo, situación que desde el punto del vista del suscriptor es totalmente transparente. De este modo, si un publicador falla, el suscriptor seguirá viendo la copia proveniente del publicador de respaldo.

La implementación de esta capacidad se basa en la calidad de servicio *OWNERSHIP* de DDS. De acuerdo a esta política de calidad de servicio, si existen múltiples publicadores publicando el mismo tópico, se puede establecer una prioridad entre ellos, de modo que mientras el publicador de mayor prioridad se encuentre activo, se descartan en el suscriptor las publicaciones de los de menor prioridad.

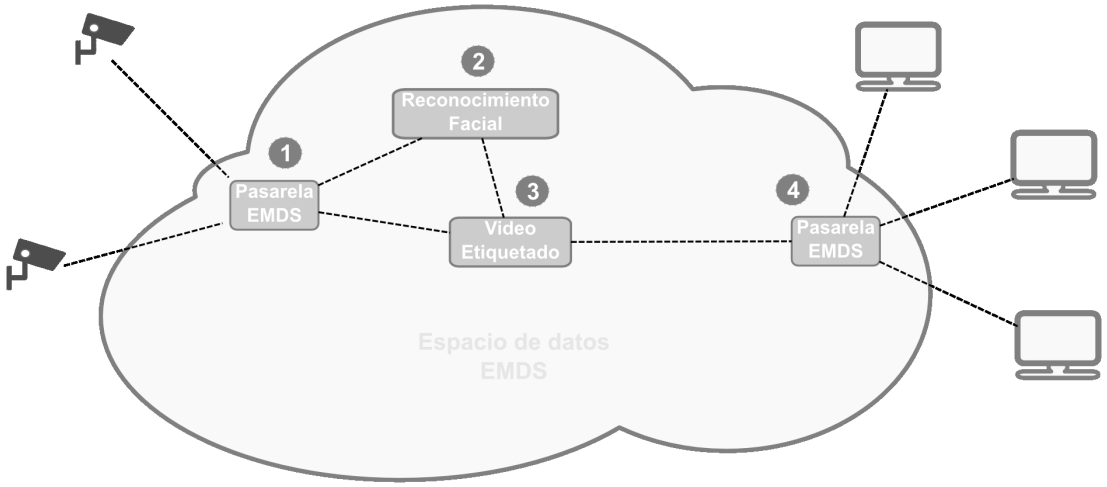


Figura 3.12: Escenario de vídeo-vigilancia basado en EMDS.

3.5. Casos de uso y servicios multimedia EMDS en el *Cloud*

Para destacar las ventajas del uso de EMDS frente a otras alternativas, en esta sección se procederá a realizar la descripción una aplicación de procesamiento multimedia distribuido basada en EMDS.

La Figura 3.12 muestra el escenario elegido para demostrar las bondades de EMDS. Concretamente, el escenario elegido muestra una aplicación de vídeo-vigilancia que analiza, procesa y visualiza flujos de vídeo. Se muestran los siguientes componentes:

1. El vídeo es capturado por cámaras IP. Dichas cámaras proporcionan acceso al mismo vídeo por medio de flujos RTP. Dichos flujos se reciben por pasarelas EMDS que se encargan de mapearlos como tópicos DDS al espacio de datos de EMDS para su posterior procesamiento. A partir de este momento, los flujos EMDS pueden ser descubiertos y procesados por diferentes *prosumers*.

2. En un segundo paso, se ha instalado un servicio EMDS dedicado al reconocimiento facial. Dicho servicio, se suscribe a flujos EMDS detecta posibles rostros faciales presentes en ellos y estima la identidad de los rostros detectados. Como resultado a dicho proceso de análisis devuelve la posición dentro del vídeo y la identidad del candidato, así como la localización de la cámara capturadora. Los resultados son publicados como tópicos DDS que pueden ser utilizados por otra aplicación que soporte DDS (*e.g. logger* de eventos).

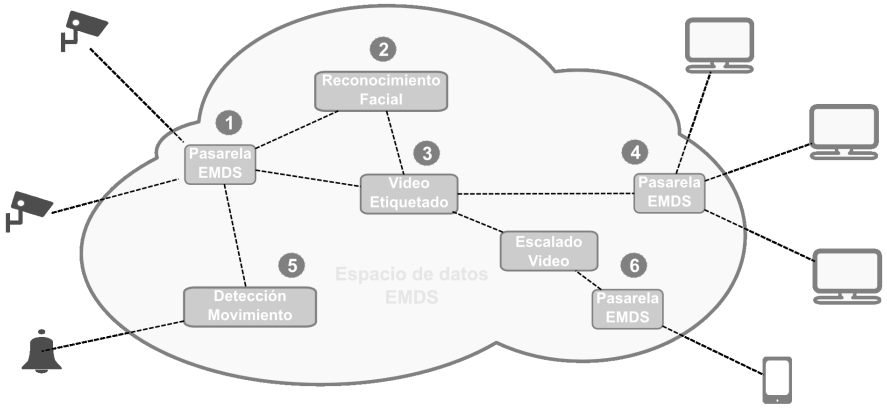


Figura 3.13: Escenario de vídeo-vigilancia basado en EMDS: nuevos servicios.

3. Otro servicio *EMDS* es desplegado. Dicho servicio está orientado a etiquetar y marcar con recuadros los rostros detectados dentro del flujo de vídeo. Para ello tiene como entrada los flujos de vídeo originales y las posiciones y tamaño de los rostros detectados, como salida genera un flujo de vídeo con toda esa información sobreimpresa.

4. Finalmente, los flujos procesados y transformados son publicados como flujos RTP y pueden ser visualizados por dispositivos que soporten dicha tecnología. Para ello basta acceder a la *URL* donde el servicio encuentre publicando dichos flujos.

Uno de las ventajas que supone el uso de EMDS frente a otras arquitecturas es que permite extender un sistema ya desplegado de una manera transparente y con una intervención humana mínima. Esto es debido a que los servicios basados en EMDS pueden descubrir automáticamente los flujos que están siendo publicados, sus características y a su vez publicar nuevos flujos derivados que serán descubiertos de igual manera. Los beneficios de esta aproximación se ponen de manifiesto en los siguientes escenarios:

- Se desea añadir un servicio que detecte automáticamente movimiento en las cámaras situadas en el exterior (*outdoor)*, y que en caso afirmativo accione una alarma.

- Se desea añadir soporte a nuevos dispositivos de capacidades reducidas (e.g. un teléfono móvil). Dichos dispositivos están interesados en flujos de vídeo con rostros etiquetados, pero a una resolución menor acorde con este tipo de dispositivos.

Para el primer caso (detección de movimiento), se lanzará un servicio de análisis EMDS que ante la detección de movimiento, invocará una determinada acción a elección del administrador. Dicho servicio de análisis, se suscribirá a los flujos EMDS que contengan la palabra clave *outdoor*, para así realizar el análisis en cámaras exteriores.

En el segundo caso, el relacionado con añadir soporte a un nuevo tipo de dispositivos, se ejecutará un servicio de transformación que genere flujos de vídeo derivados de acuerdo al nuevo tipo de dispositivos al que se desea dar soporte (en este caso reducir la resolución). Para este caso además se ha optado con añadir una nueva pasarela de salida, que exporte los flujos EMDS generados en este escenario, aumentando así el número posible de clientes posibles que pueden acceder a los flujos de vídeo generados en EMDS.

Como se observa en la Figura 3.13, basta con iniciar los servicios EMDS correspondientes dentro del espacio de datos común, para así descubrir los flujos multimedia disponibles y comenzar su procesamiento y publicación de flujos derivados. Esto contrastaría radicalmente con una aproximación cliente, servidor, donde la inclusión de un nuevo servicio distribuido, supondría la reconfiguración por parte de administrador de todos los servicios que utilicen los flujos generados por este nuevo servicio.

Finalmente, destacar que como puede observarse en este escenario de ejemplo, EMDS es un buen candidato para la creación de servicios de procesamiento multimedia en la nube (Cloud Computing). De acuerdo a esto, mientras los usuarios productores y consumidores de flujos multimedia se situarían "fuera de la nube", toda la gestión interna y procesamiento de los flujos se realizaría de forma escalable *en la nube* realizando la comunicación interna mediante *EMDS*.

3.6. Validación de la arquitectura

Para evaluar el sistema propuesto es crítico comprobar que éste cumple los exigentes requisitos que tiene el procesamiento de flujos multimedia en tiempo real, especialmente en relación a los elevados *bit-rates* que pueden llegar a consumir y a la sensibilidad a los retardos involucrados.

Con este objetivo, en esta sección se procederá a realizar una evaluación de las capacidades de transmisión de datos de EMDS. Para ello se han realizado una serie de experimentos que miden el impacto en los parámetros de red determinantes en la transmisión de flujos multimedia en tiempo real: retardo, *jitter* y ancho de banda consumido.

Software	Descripción	Version
Gstreamer	*Framework* para la construcción de aplicaciones multimedia	0.10.36
RTI DDS	Middleware Data Distribution Service (DDS)	4.5d
libav	Librerías de manejo de formatos multimedia	0.7.3
librabbitmq	Librería cliente AMQP	0.0.1

Tabla 3.1: Software empleado para la experimentación.

Nombre	Duración	Resolución	Bitrate (kb/s)	Tamaño (bytes)
video1-fullhd	00:09:56.45	1920x1080	3519	262430174
video1-hd	00:09:56.45	960x720	2021	148475128
video1-sd	00:09:56.45	480x360	304	35129438
video2-fullhd	00:01:21.68	1920x1080	3216	34192016
video2-hd	00:01:21.68	960x720	1226	14291909
video2-sd	00:01:21.68	480x360	325	5427003

Tabla 3.2: Vídeos utilizados en la experimentación.

3.6.1. Descripción del escenario experimental

Para la evaluación de EMDS se ha optado por la realización de prototipos. Dichos prototipos han sido desplegados en un entorno de red controlado compuesto por tres ordenadores Core i5, con el sistema operativo GNU/Linux 10.10 e interconectados por un *switch* Gigabit Ethernet. Las versiones del software y librerías empleados pueden encontrarse en la Tabla 3.1. El retardo medio entre los distintos equipos (*round trip time* obtenido con *ping*) es 0.15 ms.

La experimentación realizada consiste en medir los retardos y ancho de banda empleado mientras se realiza la transmisión de un flujo de vídeo a través de los distintos sistemas considerados (Figura 3.14). Se ha elegido un flujo de vídeo por su gran demanda de recursos. Además de EMDS, como sistemas de referencia se han desarrollado dos prototipos adicionales: un sistema cliente/servidor basado en RTP [165] y otro publicación/suscripción basado en AMQP [184]. Para favorecer la igualdad entre los prototipos, éstos utilizan los mismos componentes para la lectura de ficheros multimedia, codificación y medición.

La medición de los retardos se ha realizado por medio de ficheros de trazas (Figura 3.14). Para ello, en cada extremo se captura una marca de tiempo: una correspondiente al instante justo antes de enviar cada fragmento del flujo de vídeo, y

Nombre	Tamaño Frame (bytes)		
	Avg	Max	Min
video1-fullhd	18332.5	796577	68
video1-hd	10527.8	346193	28
video1-sd	1584	61896	16
video2-fullhd	18591.7	274699	791
video2-hd	7323.22	166672	180
video2-sd	1882.99	54651	27

Tabla 3.3: Tamaños de paquete de los vídeos empleados.

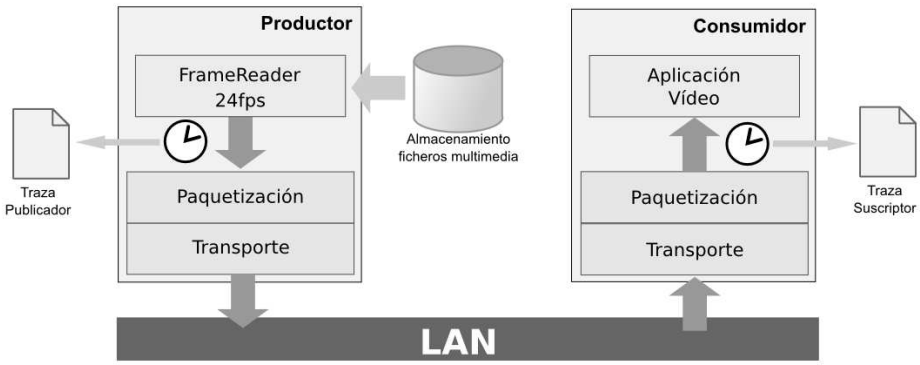


Figura 3.14: Entorno experimental de evaluación.

otra cuando el fragmento está disponible para ser enviado al decodificador. Los fragmentos de vídeo empleados se han codificado previamente, primero para eliminar los retardos de codificación en la evaluación de los sistemas y segundo para utilizar exactamente el mismo flujo de vídeo en todas las campañas. Para estudiar la influencia del tamaño de los paquetes empleados, se han considerado tres resoluciones del mismo vídeo: fullhd (1080p), hd (720p) y estándar (360p). En los experimentos se ha utilizado el codificador H.264. En las Tablas 3.2 y 3.3 se proporcionan los detalles de codificación del mismo.

Un último aspecto a tener en cuenta es que todos los ordenadores se encuentran sincronizados por medio de Precision Time Protocol (PTP), un sistema de sincronización de relojes más preciso que Network Time Protocol (NTP) [128]. De este modo se reduce la influencia de la posible desincronización de los relojes en las mediciones. Así, en los experimentos en los que se necesitan mediciones del retardo extremo a extremo, se han sincronizado los relojes utilizando este protocolo, obteniendo una precisión en la sincronización de $\pm 10\mu$-segundos, cantidad despreciable en comparación con los retardos típicos en los experimentos evaluados.

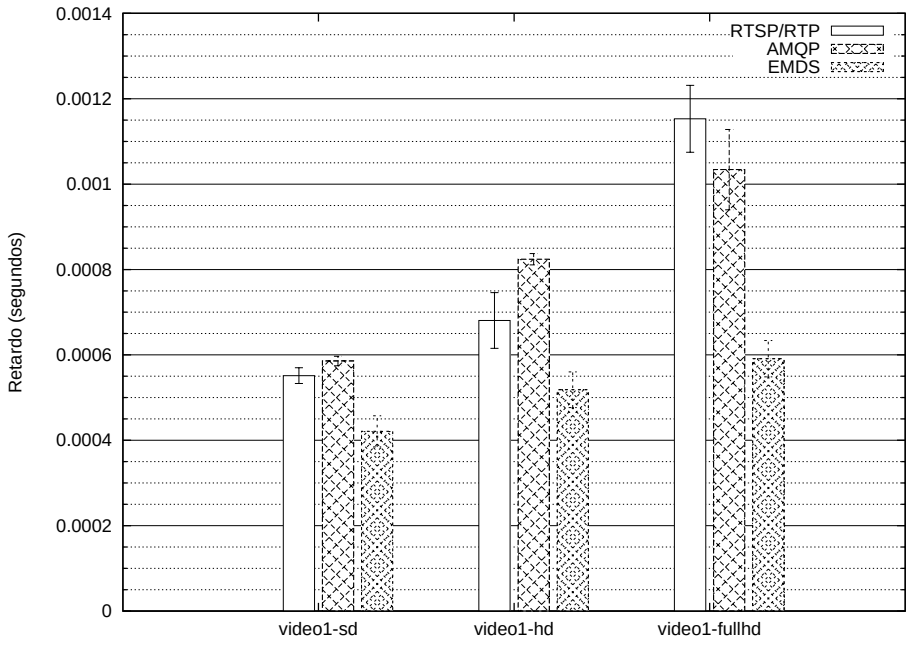


Figura 3.15: Retardo medio.

3.6.2. Estudio del retardo punto a punto

En primer lugar se ha realizado una campaña de experimentos orientados a medir el impacto en el retardo entre un productor y un consumidor de flujos multimedia. Para este experimento se han considerado los prototipos de publicación/suscripción EMDS, AMQP y el prototipo de cliente/servidor basado en RTP.

Se observa en las Figuras 3.15 y 3.16 que el retardo y el *jitter* introducido por el prototipo implementado usando EMDS es menor que en los otros dos casos.

En el caso del prototipo AMQP, el retardo se introduce fundamentalmente en el *broker* implicado, el cual debe recibir y redirigir cada fragmento del flujo a los correspondientes suscriptores. En el caso de la aproximación cliente/servidor la estrategia de fragmentación de datos de RTP es relevante. Debido a la alta resolución de los vídeo empleados, cada cuadro codificado se fragmenta en múltiples paquetes RTP que deben ser reensamblados en el destino, hecho que queda de manifiesto al observarse retardos mayores cuanto mayor es la resolución.

3.6.3. Eficacia en el despliegue de servicios. Impacto en el retardo

En este experimento se pretende demostrar la eficiencia en términos de retardo de la aproximación empleada por EMDS frente a un sistema de referencia imple-

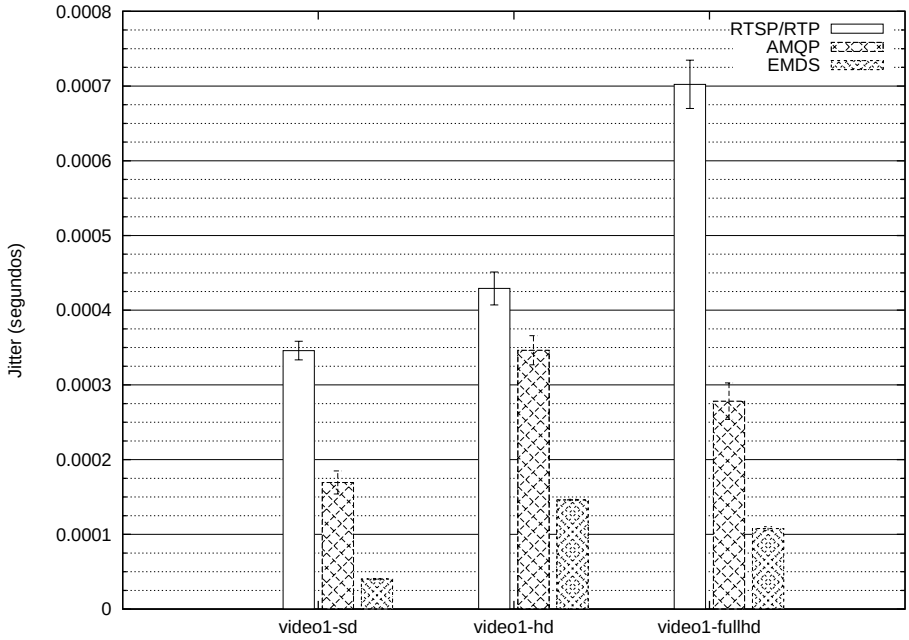


Figura 3.16: *Jitter* medio.

mentado usando AMQP al desplegar un servicio multimedia. No se ha considerado para esta campaña de experimentos el sistema basado en cliente/servidor, debido a que este paradigma no permite el despliegue de servicios de este modo. Para ello se ha desplegado un publicador de vídeo, un subscriptor y un servicio de vídeo (Figura 3.17).

El servicio de vídeo desplegado como ejemplo es un servicio de pasarela que publica una copia del flujo de vídeo usando otra publicación distinta, sirviendo así por ejemplo como un sistema de balanceo de cargas. No obstante, la naturaleza del servicio desplegado no es relevante para la medición del retardo, por lo que se podría haber implementado cualquier otro servicio multimedia.

En esta campaña de experimentos, se han transmitido los vídeos especificados en la Tabla 3.2 y se ha medido el retardo medio ocurrido de extremo a extremo. Para ello, en cada experimento se ha desplegado un servicio multimedia, que se suscribe a un flujo de vídeo original, y lo republica como otro flujo de vídeo. En el otro extremo, un suscriptor se suscribe a este nuevo flujo de vídeo. Para medir únicamente el impacto de EMDS, el servicio utilizado para este experimento es un servicio de repetición, para así evitar considerar el impacto de un operador en concreto.

Para la medición del retardo se han tomado marcas de tiempo de manera similar a como se ha realizado en el experimento anterior: se ha tomado el tiempo en el publicador previamente al envío de los datos al *middleware* y en el suscriptor se ha tomado la marca de tiempo justo antes de pasar los datos al decodificador multime-

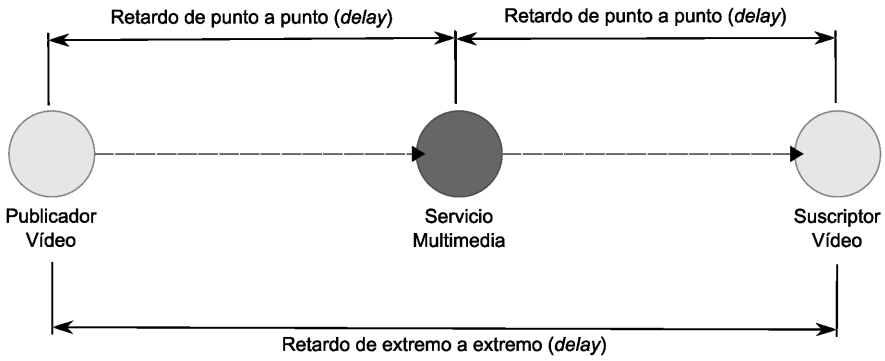


Figura 3.17: Retardo de extremo a extremo.

dia.

En la Figura 3.18 se observa que para el caso de EMDS, los retardos son significativamente menores, especialmente en el caso de utilizar transporte *BEST_EFFORT*. Esto es debido a dos factores: el primero de ellos es que AMQP depende de un *broker* por el que deben de pasar los mensajes entre el publicador y el servicio, y los mensajes desde el servicio al suscriptor. Además este hecho puede llegar a ser más apreciable en escenarios donde el número de servicios intermedios incremente, ya que serán necesarios múltiples pasos por el *broker*. En segundo lugar, hay otro factor que influye en el retardo, y es que AMQP usa el protocolo orientado a conexión TCP, mientras que EMDS utilizó datagramas User Datagram Protocol (UDP).

Se puede observar por tanto que en la provisión de servicios EMDS introduce un retardo menor que AMQP, siendo por tanto más apropiado para escenarios donde los requisitos sean más exigentes.

3.6.4. Impacto del uso de comunicaciones fiables

Una de las ventajas de EMDS es el uso del estándar DDS como núcleo de comunicación. Este estándar define una gran variedad de políticas de calidad de servicio configurables cuando se compara con otros estándares de la bibliografía. En este apartado se evalúa el impacto causado en el retardo por la inclusión de la política de calidad de servicio *RELIABILITY* del estándar DDS.

Para este experimento se ha transmitido la secuencia *video1* desde un publicador a un suscriptor EMDS y se ha medido el retardo incurrido desde que se transmite cada paquete hasta que se recibe en el otro extremo. En este experimento se han considerado las tres resoluciones disponibles para la secuencia *video1* repitiendo la ejecución un total de 5 veces. La Figura 3.19 muestra los retardos medios obtenidos

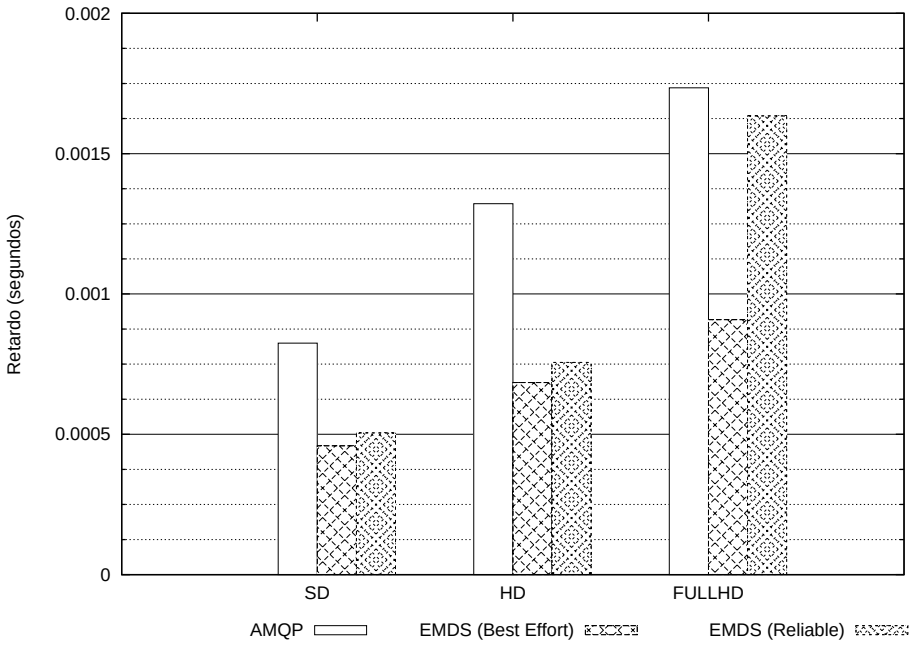


Figura 3.18: Retardo al proporcionar un servicio.

para cada resolución y calidad de servicio. En este caso, el tamaño de fragmento de mensaje utilizado es el valor por defecto (8K).

Se observa, como cabía esperar, que en el caso de utilizar la calidad de servicio *BEST_EFFORT* el retardo introducido es menor que *RELIABILITY* para todas las resoluciones. Se observa además, que a mayor resolución la influencia de la entrega fiable en el retardo medio adquiere mayor importancia debido al procesamiento extra necesario y a la retransmisión de paquetes, pero siempre manteniéndose siempre dentro de unos valores razonables para el caso de comunicaciones multimedia (alrededor de 0.5 ms). Finalmente, destacar que la varianza del retardo obtenido ha sido razonablemente pequeña. A tenor de los resultados, podemos concluir que ambos tipos de comunicaciones son válidos para entornos multimedia, pudiendo elegir por tanto el usuario uno u otro en función de sus requerimientos (fiabilidad o mínimo retardo).

3.6.5. Evaluación del impacto de la fragmentación

En este experimento se estudia la influencia de la fragmentación en las prestaciones de EMDS.

El transporte en EMDS permite fragmentar los mensajes grandes en submensajes, los cuales se reensamblan en el suscriptor. La fragmentación implica un incre-

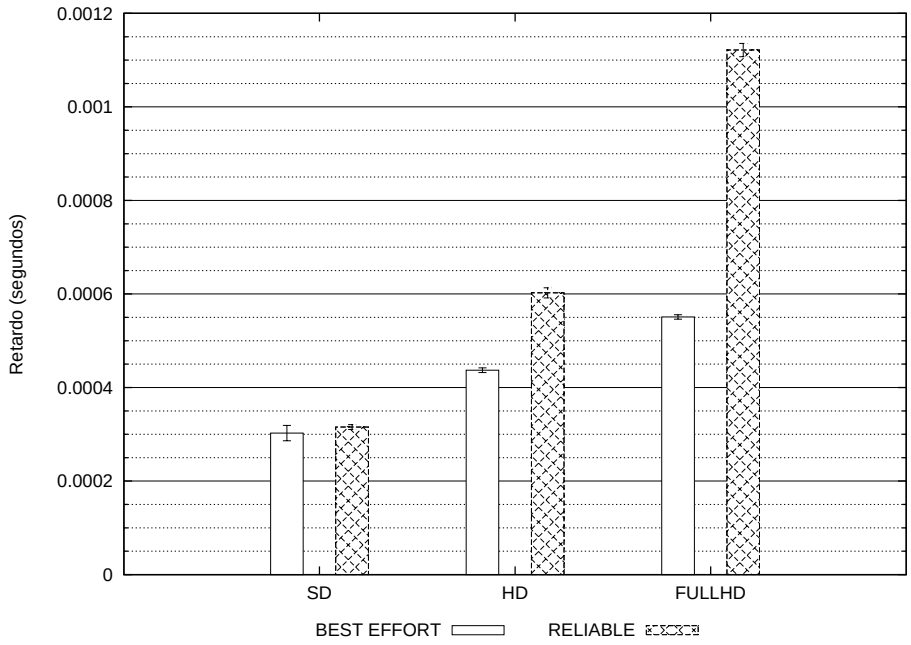


Figura 3.19: Retardo de extremo a extremo: *best-effort* vs. *reliable*

mento del tráfico debido a las cabeceras de cada submensaje. Además de un incremento en el tráfico, también conlleva un incremento en el retardo, debido al coste por el reensamblado de paquetes en el receptor. En esta sección se mide este impacto.

Para esta campaña de experimentos se utilizan las secuencias video1 y video2, en las tres resoluciones (*SD*, *HD* y *FULLHD*). Se ha medido en cada caso el tráfico total generado así como el retardo medio por muestra. Además, para caracterizar su influencia se consideran diferentes tamaños de fragmentos, concretamente 2K, 4K, 8K, 16K, 32K y 64K.

Los *hosts*, al igual que en experimentos anteriores, disponen de procesador Intel i5 y tarjeta Gigabit ethernet, con un *round trip time* de aproximadamente 0.15 ms. El tráfico generado se calcula como el agregado de los tamaños de todos los paquetes *RTPS* enviados.

En la Figura 3.20 se muestra la influencia del tamaño del fragmento respecto al tráfico total generado. Se observa que para resoluciones pequeñas (*SD*) el fragmento no es relevante en el tráfico generado, esto se debe a que el número de muestras que se fragmentan es reducido debido a que el tamaño de cada cuadro codificado rara vez excede los 2K. Sin embargo, se observa que para resoluciones mayores (*HD* y *FULLHD*) la cantidad de tráfico aumenta (de forma más apreciable en *FULLHD*) cuanto menor es el fragmento. Esto se debe a que mayores resoluciones implican más bytes por cuadro, es decir mayor probabilidad de fragmentación.

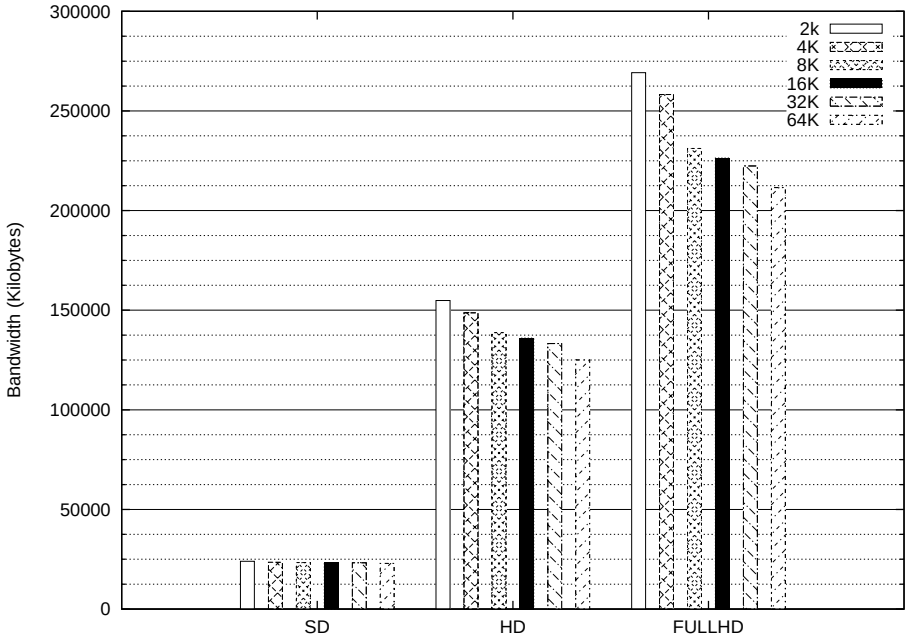


Figura 3.20: Fragmentación vs. tráfico generado.

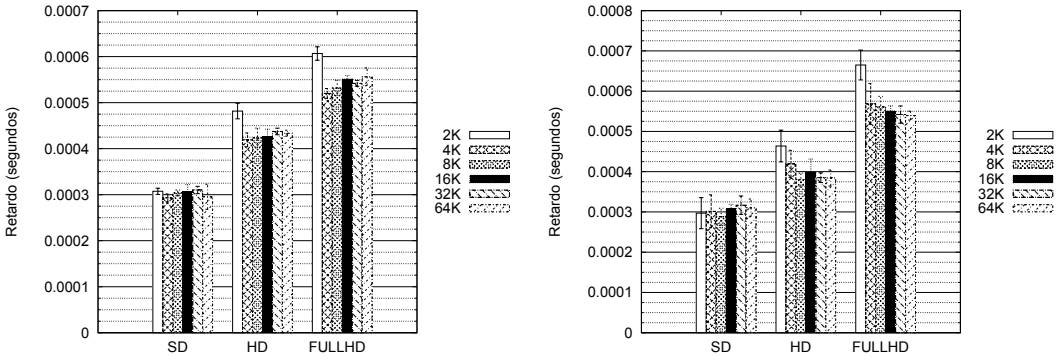
En el siguiente experimento se evalúa el impacto del tamaño de los fragmentos en el retardo. En la Figura 3.21 se representan los retardos medios por paquete para la secuencia video1 (Figura 3.21a) y video2 (Figura 3.21b) respectivamente. Al igual que en el experimento anterior y por la misma razón, para la resolución más baja (SD), se observa que el tamaño de fragmento no tiene un impacto significativo en el retardo. Ahora bien para resoluciones mayores se observa que fragmentos de 2K implican un mayor retardo, lo que puede estar relacionado con la necesidad de enviar más de un mensaje para ciertos cuadros con mucha información.

3.6.6. Evaluación del impacto de la seguridad

EMDS incluye mecanismos de seguridad, tales como privacidad, autenticación e integridad (ver Sección 3.2.5). Sin ánimo de exhaustividad, en esta Sección se procederá a evaluar el impacto que la seguridad tiene en el retardo.

Para este experimento se ha transmitido la secuencia *video1* de 10 minutos de duración desde un publicador a un suscriptor EMDS (los resultados se obtienen tras 5 repeticiones). Para eliminar el retardo que la red puede introducir, tanto publicador como suscriptor se encuentran en el mismo ordenador. En este caso se opera en *BEST_EFFORT*.

Se evalúan tres variantes de cifrado: total, parcial y sin cifrado. En la primera, se



(a) Secuencia *video1* (b) Secuencia *video2*

Figura 3.21: Retardo para diferentes tamaños de fragmento: secuencia *video1* 3.21a y *video2* 3.21b.

cifran todos y cada uno de los paquetes. En la segunda, se consideran únicamente los *keyframes*, es decir los paquetes que son necesarios para decodificar los otros, y finalmente la tercera variante consiste en no cifrar ninguno de los paquetes.

Los resultados (Figura 3.22) muestran que el retardo introducido en la primera variante es muy elevado, llegando a un factor de aproximadamente 2.5 en el caso de la máxima resolución (FULLHD) frente al cifrado parcial. Este hecho se debe fundamentalmente al mayor tamaño que tienen los cuadros de tipo no *keyframe* en flujos de mayor resolución. No obstante, se observa que la aproximación de cifrado parcial propuesto en EMDS supone una reducción en retardo del orden de 0.6 ms en el caso de mayor resolución, lo que lo hace apropiado para escenarios donde los retardos sean críticos y a la vez se necesiten garantías de seguridad.

Además del cifrado, se ha medido la influencia del mecanismo de autenticación mediante firmado HMAC en el retardo. Se observa en la Figura 3.22, que la influencia del retardo es también mínima, obteniendo retardos similares al cifrado total en definición SD, y algo menores en el caso de resoluciones mayores (HD y FULLHD).

3.7. Conclusiones

En este capítulo se ha propuesto EMDS una arquitectura distribuida para la provisión y el despliegue de servicios multimedia distribuidos.

EMDS propone el uso de una aproximación de publicación/suscripción centrada en datos para el intercambio de flujos multimedia entre los distintos nodos de procesamiento.

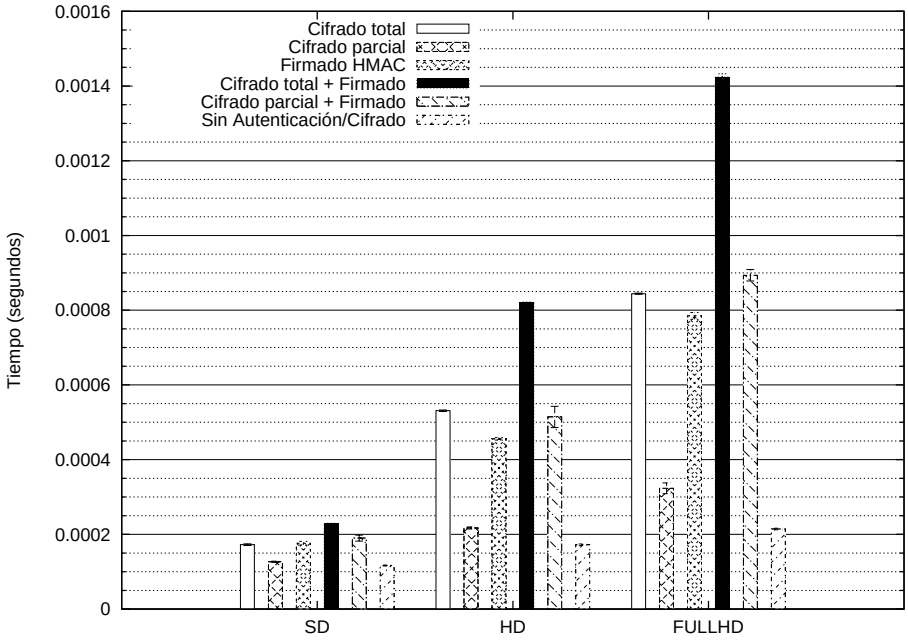


Figura 3.22: Impacto del cifrado en el retardo.

EMDS se diseña sobre el estándar DDS como *middleware* de publicación/suscripción, lo que permite obtener una serie de ventajas frente a otras alternativas como AMQP o otras aproximaciones clásicas basadas en cliente/servidor.

De acuerdo a los requisitos especificados (ver Sección 3.1) se obtienen las siguientes ventajas:

Distribuido. La aproximación publicación/suscripción permite crear sistemas desacoplados y distribuidos de una manera sencilla.

Mínima latencia. EMDS utiliza el estándar DDS que está especialmente concebido para satisfacer requisitos de tiempo real. El uso de calidades de servicio y la no inclusión de *brokers* intermedios reducen el retardo entre publicadores y suscriptores, tanto en interacciones que impliquen transmisiones de flujos punto a punto como en escenarios que impliquen la provisión de un servicio.

Diversos media y formatos. Los tipos de datos definidos por EMDS están diseñados para no depender de los *codecs* o contenedores específicos empleados, lo que permite añadir soporte a nuevos formatos y a satisfacer necesidades heterogéneas –puede que cambiantes en el tiempo– de una manera sencilla. Además, EMDS proporciona mecanismos para el despliegue de réplicas de flujos multimedia –puede que en distintos formatos al original– proporcionando una mayor robustez al sistema.

Extensible. EMDS permite el despliegue no disruptivo de servicios *en operación* lo que hace que el sistema sea extensible.

Descubrimiento. Los suscriptores EMDS permiten descubrir automáticamente los flujos multimedia existentes así como sus características, lo que permite a los suscriptores seleccionar a qué flujos suscribirse. Esto convierte a EMDS en un entorno altamente flexible.

Escalabilidad. El desacoplo entre los productores y consumidores de flujos multimedia en EMDS permite el despliegue de servicios a demanda, aumentando así la escalabilidad del sistema.

Integración y uso de estándares. EMDS está basado en tecnologías estandarizadas como EMDS y JSON, lo que facilita la integración con otros sistemas desplegados. Además, EMDS permite la construcción de pasarelas que permiten la integración de EMDS con otros servidores multimedia.

Seguridad. EMDS proporciona mecanismos para autenticar, garantizar integridad y confidencialidad de los flujos multimedia publicados. Estos mecanismos no son excluyentes con usar protocolos seguros en las capas inferiores de la arquitectura de red.

4

Monitorización de recursos en entornos distribuidos

El procesamiento, análisis y distribución de flujos multimedia son por lo general tareas muy demandantes de recursos. Por ello, es bastante habitual diseñar sistemas distribuidos en los que dichas tareas sean repartidas en múltiples nodos.

En el despliegue y operación de todo sistema distribuido, conocer en cada momento los recursos disponibles y su nivel de utilización puede ser decisivo para mejorar el rendimiento y la robustez del sistema. Para el administrador es del máximo interés tener información acerca de la disponibilidad de recursos, la distribución de la carga entre los mismos y en su caso, si existe o hay peligro potencial de sobrecarga.

Uno de los mayores inconvenientes en el procesamiento distribuido de flujos multimedia es que una incorrecta gestión de los recursos puede acarrear –por ejemplo– incrementos de retardo o *jitter*, lo que significará una degradación de calidad y en definitiva la no verificación de requisitos del sistema. Así por ejemplo, si un sistema distribuido de detección de objetos (*tracking*) de vídeo presentara sobrecarga en alguno de sus nodos, podría causar retardos no tolerables e incluso detecciones falsas. Poder añadir redundancia (nuevos nodos) al sistema *on-line* (es decir, sin interrupción del servicio) sería deseable para paliar esta potencial debilidad. Para ello, determinar cuándo y cómo se ha de realizar esta gestión es de vital importancia, y es aquí donde disponer de un sistema de monitorización fiable y eficiente se hace necesario.

Los hechos expuestos anteriormente, evidencian la importancia de disponer de un completo y eficaz sistema de monitorización en entornos de procesamiento y distribución multimedia, especialmente si existen requerimientos de tiempo real. En el presente capítulo se abordará este tema y se obtendrá como resultado un sistema de monitorización que satisfaga las necesidades de tiempo real de un sistema

distribuido de procesamiento multimedia, como son la autenticidad de los datos de monitorización, la escalabilidad, la robustez ante pérdidas y un impacto mínimo en el ancho de banda requerido por el sistema. Gracias a estos sistemas se puede disponer de información acerca de de los recursos disponibles en cada nodo que permitirá a su vez diseñar estrategias eficaces de distribución de tareas que satisfagan los requerimientos en cada escenario en particular. Concretamente, conocer el estado global del sistema permite al administrador adoptar estrategias de gestión de recursos que abarquen desde el balanceo de carga entre los nodos disponibles [192], hasta la minimización de número de nodos necesarios para reducir el consumo energético (consolidación) [34, 37, 195].

En este capítulo se propone una arquitectura para la monitorización de sistemas multimedia distribuidos. De nuevo, la estrategia seguida ha sido aprovechar los beneficios que la aproximación *data-centric* y el paradigma publicación/subscripción pueden aportar a la resolución de este problema.

El capítulo se ha organizado de la siguiente manera. En la Sección 4.1 se describen los requisitos que debe cumplir el sistema de monitorización. En la Sección 4.2 se explica el diseño del sistema propuesto. A continuación, en la Sección 4.3 se proporcionan detalles sobre la implementación llevada a cabo. En la Sección 4.4 se incluye la validación realizada y finalmente, el capítulo concluye resumiendo las principales conclusiones obtenidas (Sección 4.5).

4.1. Requerimientos

Antes de abordar el diseño es necesario definir los requisitos que debe cumplir el sistema propuesto, que en este caso tiene como objetivo la monitorización de un entorno de distribución multimedia. A continuación se definen las características más relevantes exigibles:

Tolerancia a fallos. El sistema debe ser tolerante a fallos. Esto significa que ante el fallo de un nodo, el sistema debe seguir funcionando con normalidad y en el peor de los casos el error debe quedar confinado exclusivamente en el nodo afectado.

Descubrimiento. El sistema propuesto debe permitir añadir nuevos nodos al sistema de una manera transparente, intentando evitar la configuración manual o mediante ficheros de configuración. Dicho descubrimiento debe ser bidireccional, permitiendo tanto a los nodos que generan información de monitorización como a los que la utilizan descubrirse mutuamente.

Seguridad. Un sistema de monitorización necesita disponer de mecanismos que permitan garantizar de forma fehaciente la identidad del origen de los datos de

monitorización, de modo que no se permita la suplantación de un *host*, ni por consiguiente la publicación de datos de monitorización que no sean ciertos. En otros casos además puede ser un requisito que los datos de monitorización sean inaccesibles a usuarios no autorizados.

Integridad. Un sistema de monitorización debe de garantizar la integridad de los datos de monitorización, evitando así que la información de monitorización pueda ser manipulada o alterada respecto a cómo fue publicada.

Filtrado. No todas las aplicaciones interesadas en monitorizar *hosts* remotos necesitan recibir toda la información disponible. En algunos casos, incluso esto es no deseable, para así evitar sobrecargas y/o consumo de recursos innecesarios. Es por tanto deseable que el sistema de monitorización permita el filtrado de información, para que se pueda controlar los consumidores/receptores de la información.

Acceso histórico. Un sistema de monitorización, no solo debe proporcionar la información actual, sino que además debe disponer de mecanismos para acceder al histórico, esto permitirá observar tendencias y hacer predicciones futuras de uso (*forecasting*).

Control. En la mayoría de escenarios es suficiente con sensores pasivos que recolecten las estadísticas de uso para enviarlas a los nodos agregadores de dicha información. Sin embargo, en ciertos ocasiones es necesario tener control remoto sobre estos nodos para reconfiguración (*e.g.* aumentar la frecuencia de recolección de datos). Es por tanto deseable que el sistema de monitorización disponga de algún mecanismo para controlar remotamente los sensores.

Estandarización. Los sistemas de monitorización no tienen mucha utilidad *per se*, sino que por lo general se usan para recopilar información tal que otro servicio –como una consola de visualización o un distribuidor (también denominado balanceador) de carga– la utilice. Es por esto que el uso de tecnologías, protocolos y formatos estandarizados es un requisito importante para facilitar la interoperabilidad de dichos sistemas. El uso de estándares permite aumentar las capacidades de integración de la herramienta con otras aplicaciones.

Extensibilidad. Un sistema de monitorización debe incluir por defecto soporte para monitorizar los recursos más básicos como son CPU, memoria o carga del sistema. No obstante, en algunos escenarios es necesario monitorizar algunos recursos como el estado de un servicio o una aplicación. Por ello, es deseable poder añadir mecanismos que permitan la monitorización de nuevos recursos. Estos nuevos recursos se deben poder integrar con facilidad y sin interferir con los ya existentes.

Escalabilidad Los sistemas de monitorización están compuestos por multitud de nodos, que a su vez contienen multitud de recursos que deben de ser monitorizados

y que además pueden incrementar su número a lo largo del tiempo. Por este motivo, es de vital importancia que el sistema propuesto sea escalable.

4.2. DARGOS

Los sistemas comentados en el Capítulo 2 y más concretamente en la Sección 2.3 representan el estado del arte actual en monitorización de recursos en entornos de red. Por lo general, cada sistema está especializado en un escenario en concreto, por lo que su desempeño no siempre es óptimo para todos los posibles entornos de aplicación.

Así por ejemplo, *Nagios* [124] proporciona una interfaz web indicada para monitorizar pasivamente servicios y despliegues, pero carece de las características de tiempo real deseables en sistemas que requieran una monitorización de grano fino, como por ejemplo aquellos entornos orientados a distribuir tareas en red. Respecto a la extensibilidad, *Nagios* proporciona un amplio soporte [126], pero como se ha mencionado adolece de las características deseables para monitorizar sistemas muy cambiantes.

Por otro lado, *Ganglia* [116] está orientado a entornos *Grid* y permite monitorizar recursos hardware con gran precisión. No obstante, *Ganglia* carece de herramientas que permitan filtrar la información de monitorización, pudiendo causar saturaciones innecesarias de los nodos monitorizadores (*e.g.* sobrecarga por recepción de actualizaciones demasiado frecuentes).

Otro ejemplo es *Lattice*, [29, 32] que aunque cumple la mayoría de requisitos no define –al igual que la mayoría de sistemas– mecanismos que permitan asegurar los aspectos de seguridad.

Por ello, se concluye que ninguno de los sistemas identificados satisface plenamente todas las necesidades que se consideran deseables para la monitorización de un entorno multimedia distribuido.

Como consecuencia de la ausencia de una solución genérica que satisfaga todos los requerimientos considerados en la Sección 4.1, de este Capítulo, se propone un sistema de monitorización especialmente diseñado para entornos multimedia distribuidos. En la siguiente Sección se describen las decisiones de diseño adoptadas, su motivación y los componentes que la conforman. Finalmente, para su evaluación se proponen y explican una serie de experimentos realizados que permiten validar el diseño y la implementación de la solución propuesta.

4.2.1. Diseño

Como se ha indicado, los sistemas estudiados en la Sección 2.3 cumplen parcialmente algunas de las características o requisitos mencionados, sin embargo no existe ninguno que los verifique todos.

A continuación se explican las decisiones tomadas en algunos aspectos del diseño de la solución de monitorización propuesta:

Arquitectura Uno de los requisitos que se considera imprescindible es la tolerancia a fallos. Los sistemas centralizados, normalmente presentan menos robustez que los sistemas distribuidos, debido a los problemas derivados de tener un único punto de fallo. Por este motivo, consideramos que el sistema de monitorización para el escenario que ocupa esta Tesis debe ser distribuido, maximizando así la robustez y tolerancia a fallos.

Métricas de recolección La cantidad de métricas que se pueden recolectar dentro de un sistema distribuido es muy elevado. Así por ejemplo, es posible recolectar métricas del uso de recursos hardware –como el porcentaje de uso de CPU de un determinado dispositivo– o métricas de un nivel superior de abstracción –como la carga o uso de servicios/aplicaciones concretas–. Una posible clasificación de las métricas se establece considerando la metodología de recolección de las mismas: así denominamos métricas *internas* a aquellas que deben ser medidas desde el mismo dispositivo y *externas* aquellas que son tomadas desde el exterior (*e.g.* latencia de un servicio web).

Normalmente, las métricas *internas* están íntimamente relacionadas con recursos hardware mientras que las métricas *externas* están relacionadas con el estado de una aplicación o servicio (*e.g.* número de conexiones abiertas, peticiones realizadas, etc).

El principal inconveniente que presentan las métricas *internas* frente a las *externas* es que para que estén disponibles para otros dispositivos, deben de ser recolectadas y publicadas por un agente dedicado.

Los sistemas para la distribución de servicios multimedia dependen tanto de los recursos hardware disponibles (es decir de métricas internas como por ejemplo la carga de CPU, el uso de memoria y la carga de los dispositivos de entrada/salida de un determinado nodo, etc.) como de los retardos existentes entre los nodos que la componen por lo que consideramos que un sistema de monitorización para entornos multimedia debe estar basado en agentes remotos que se encarguen de recolectar información tanto local como remota para exponerla de forma eficaz al resto de nodos interesados.

Por este motivo, DARGOS se ha diseñado utilizando un despliegue basado en agentes, que permite recolectar tanto métricas internas como externas.

Estrategia de intercambio de datos. Según la entidad que inicie el intercambio de datos de monitorización, se pueden distinguir dos tipos de estrategias: la entidad interesada en el uso de los recursos realiza una petición (*pull*) o la entidad monitorizada es la encargada de enviar la información a las entidades interesadas (*push*).

Si bien cada estrategia tiene sus ventajas e inconvenientes, consideramos que en entornos distribuidos donde pueden existir múltiples entidades interesadas en los datos de monitorización es más adecuado adoptar la aproximación *push*. El uso de esta estrategia junto con comunicaciones *multicast*, puede reducir notablemente el uso de recursos necesarios. Así por ejemplo, con una estrategia *push* el número de mensajes disminuirá drásticamente ya que se eliminan los mensajes de petición de estado (generados desde las entidades interesadas) y si además se usan comunicaciones *multicast* un único mensaje de monitorización puede ser enviado a múltiples destinatarios. Sin embargo, en algunos escenarios los nodos solo necesitan recibir información de monitorización bajo demanda, por lo que la aproximación *pull* también es deseable. Por este motivo, el diseño de DARGOS contempla ambas estrategias.

4.2.1.1. Modelo de datos

Un aspecto clave en la diseño de un sistema centrado en datos (*data centric*), es el modelo de datos a emplear, es decir la definición de los tópicos junto con los tipos correspondientes. En este sentido, se ha intentado minimizar el número de tipos de tópicos presentes en el sistema, de modo que el impacto sobre el tráfico de red sea mínimo.

La aproximación de utilizar un tópico distinto para cada cada uno de los parámetros medidos en cada recurso presenta varios problemas: cada parámetro debe ser descubierto independientemente, cada parámetro debería estar acompañado de una serie de metadatos para identificar la medición (*e.g. UUID*), y cada muestra de tópico publicada llevaría asociadas cabeceras RTPS, en algunos casos incluso de mayor tamaño que la misma muestra. Además, para algunos recursos puede haber del orden de decenas de parámetros a medir lo que implicaría gestionar un número igual de tópicos por cada recurso a monitorizar. Así por ejemplo en el caso de la monitorización de una interfaz de red, pueden ser interesantes hasta 20 parámetros (bytes escritos, bytes leídos, errores, etc)

Por este motivo, DARGOS define un tópico por cada uno de los recursos que se deseen monitorizar (e.g. CPU, memoria). Esta aproximación permite agrupar to-

dos los parámetros medidos para un mismo recurso en el mismo tópico, logrando así minimizar el tráfico generado y obtener acceso a todas las estadísticas de uso de un recurso de una manera más compacta.

Todos los tópicos de sensor implementados tienen una serie de campos comunes, que permiten la monitorización de sistemas remotos de una manera precisa.

En primer lugar, cada tópico dispone de un campo de identificación que permite identificar unívocamente a cada recurso y al *host* al que pertenece.

En segundo lugar, cada muestra de tópico incluye una marca de tiempo correspondiente al instante de tiempo en el que fue realizada la medida, de modo que se pueda realizar un seguimiento del histórico de uso de cada recurso.

Finalmente, cada tópico incluye una firma *HMAC-SHA-1* (ver Sección 4.2.3.5), hecho que permite dotar al sistema de garantías de integridad y autenticación de la información.

4.2.1.2. Diagrama de clases

DARGOS ha sido diseñado para poder monitorizar de forma fácil y sencilla los recursos software y hardware de un sistema distribuido. Para hacer que esto sea posible debe de estar respaldado por una API sencilla.

En la Figura 4.1 se puede observar las clases principales de mayor relevancia que componen el sistema DARGOS. Como puede observarse se han intentado reducir al mínimo el número de clases y métodos que contiene cada una para así facilitar al usuario el desarrollo de sistemas basados en DARGOS. A continuación se describen las clases más relevantes:

Agent Corresponde con una definición abstracta de un servicio DARGOS. Contiene los métodos para comenzar o parar un servicio DARGOS.

NMA Como veremos adelante, un Node Monitoring Agent (NMA) es un agente encargado de monitorizar los recursos que son accesibles desde un nodo determinado y publicarlos en el espacio de datos de DARGOS.

NSA Un Node Supervisor Agent (NSA) es un agente encargado de suscribirse a la información de monitorización publicada por NMA. Los NSA permiten acceder a la estadísticas de monitorización provenientes de múltiples NMA por medio de varios métodos, tales como acceso por identificador, o acceso por recurso.

Sensor Un sensor es una abstracción de un sensor. Define una interfaz que contiene las operaciones deben implementar los sensores utilizados en DARGOS, tales como

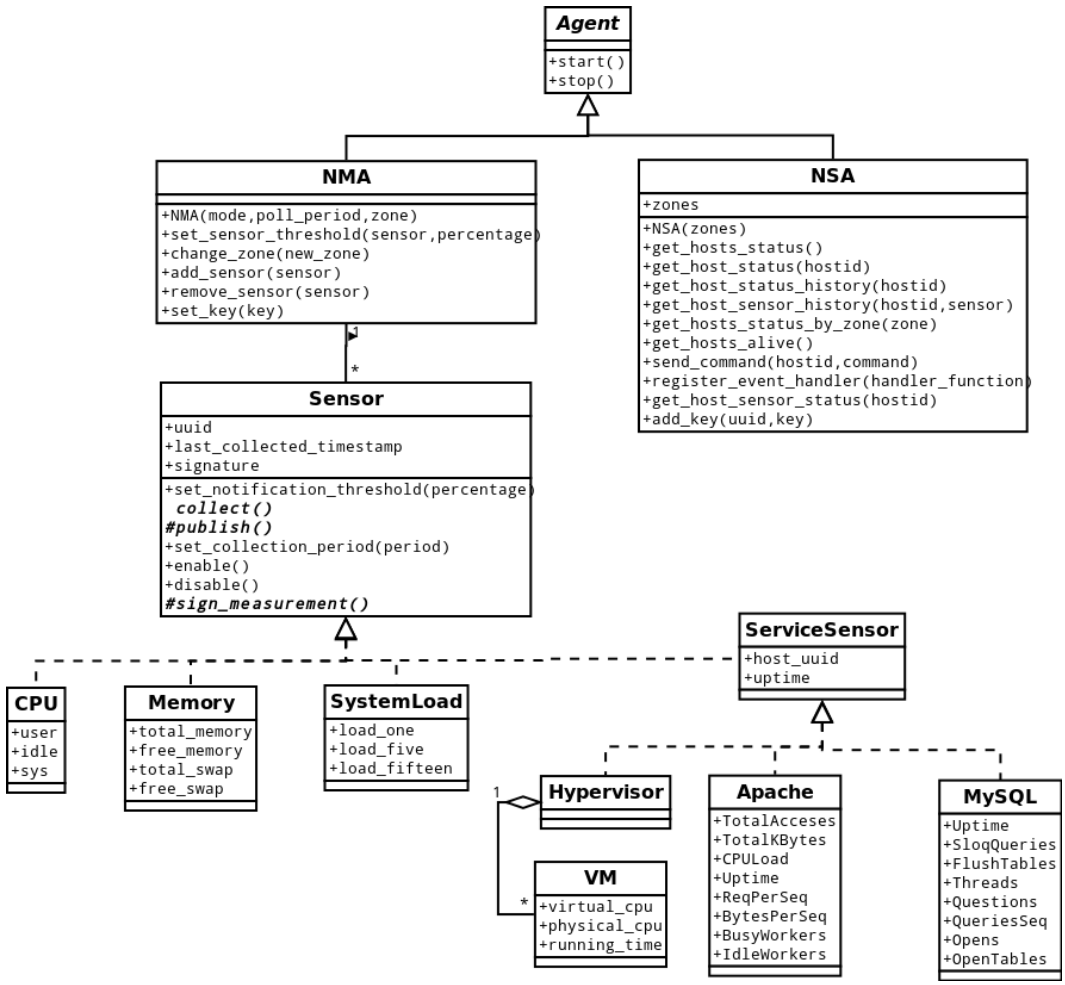


Figura 4.1: Diagrama de clases más relevantes de DARGOS.

la recolección de datos y la serialización de los mismos. Por defecto DARGOS incluye sensores para CPU, memoria, carga del sistema e interfaces de red.

ServiceSensor Los *ServiceSensor* son un tipo especializado de sensores que orientados a monitorizar las estadísticas de uso de determinados servicios. La clase ServiceSensor representa la interfaz que deben implementar los mismos. Los *ServiceSensor* pueden ser utilizado para monitorizar el estado de un servicio de base de datos, o un servidor web.

4.2.2. Agentes DARGOS

DARGOS es un sistema de monitorización distribuido cuyo despliegue se basa en el uso de agentes localizados en los distintos nodos que componen el sistema.

Como se ha mencionado anteriormente, este modelo permite recolectar tanto métricas internas (uso de hardware) como externas en los nodos.

En DARGOS se distinguen dos tipos de agentes, los Node Monitoring Agent (NMA) y Node Supervisor Agent (NSA) (Figura 4.2). Los primeros se ejecutan en aquellos nodos que se quieren monitorizar recopilando métricas internas, mientras que los segundos se encargan de recopila métricas de monitorización proveniente de múltiples NMA.

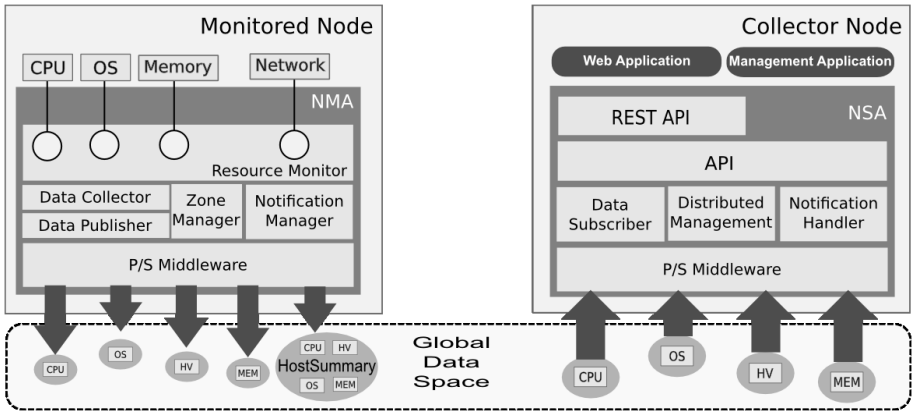


Figura 4.2: NMA y NSA.

Node Monitoring Agent (NMA). Los agentes NMA son los encargados de recopilar la información acerca de los recursos locales disponibles –tanto hardware como software– en un sistema. Para ello disponen de una serie de sensores que sondean periódicamente el uso de los recursos. La información recabada la envían a los correspondientes subscriptores de acuerdo a los requisitos de calidad de servicio establecidos por los mismos. Un NMA se puede integrar dentro de una aplicación (*e.g.* un servidor web), o bien se puede ejecutar como un proceso independiente dentro de un sistema, aumentando así la flexibilidad en el despliegue del mismo.

Node Supervisor Agent (NSA). Los NSA son los encargados de recopilar la información de monitorización de recursos remotos que esté siendo publicada. Para ello especifican la información de monitorización en la que están interesados y los requisitos de entrega que debe cumplir la información de monitorización recibida. Los NSA de DARGOS tienen las siguientes características diferenciales que no se encuentran en otros sistemas de monitorización disponibles en la bibliografía:

- No es un mero subscriptor pasivo, sino que puede definir reglas acerca de las subscripciones a monitorizar. Por ejemplo, puede seleccionar los recursos de

los que quiere recibir información (CPU, memoria, etc), la frecuencia de actualización (*i.e.* cada cuánto quiere recibir dicha información) y si la recepción de las publicaciones debe cumplir algún requisito especial (*i.e.* reglas de filtrado). Así por ejemplo, un subscriptor podría indicar que solo desea recibir actualizaciones acerca del uso de CPU cada minuto, o cuando la carga de CPU supere cierto umbral.

- Además de las métricas del uso de recursos, los NSA pueden recibir otros eventos asíncronos –como alarmas–. Para ello su diseño permite instalar unos manejadores específicos que posibilitan la automatización de tareas de gestión o notificación.
- Por último, los NSA permiten igualmente invocar procedimientos remotos en un NMA, de modo que también tienen capacidades de administración.

Respecto al despliegue de NSA, al igual que los NMA, pueden ser desplegados como procesos independientes o integrados dentro de una aplicación, ya que –como se explic ará más adelante– disponen de interfaces para acceder

a dicha información en ambos escenarios.

Una vez que se han descrito los agentes de DARGOS, a continuación se procederá a la descripción de los módulos que conforman estos agentes.

4.2.3. Componentes principales

Cada entidad DARGOS se compone de los siguientes módulos: comunicación, recolección de datos, control, consulta y seguridad. En esta Sección se describirá cada uno de ellos. Adicionalmente, en la Figura 4.3 se muestra gráficamente la relación entre ellos.

4.2.3.1. Módulo de comunicación

De acuerdo al Capítulo 2.3, la mayoría de los sistemas de monitorización existentes están basados en paradigmas arquitectónicos centralizados y modelo de comunicación *pull*. Como ya se comentó, este tipo de sistemas tienen problemas de escalabilidad, ya que el número de peticiones a realizar aumenta proporcionalmente con el número de nodos a monitorizar, aumentando por tanto el tiempo necesario para obtener una actualización completa de todo el sistema.

Para mitigar este problema, sistemas como *Ganglia* [116] o *Lattice* [29] proponen una solución basada en el uso de un modelo de comunicación *push*. Tanto *Ganglia*

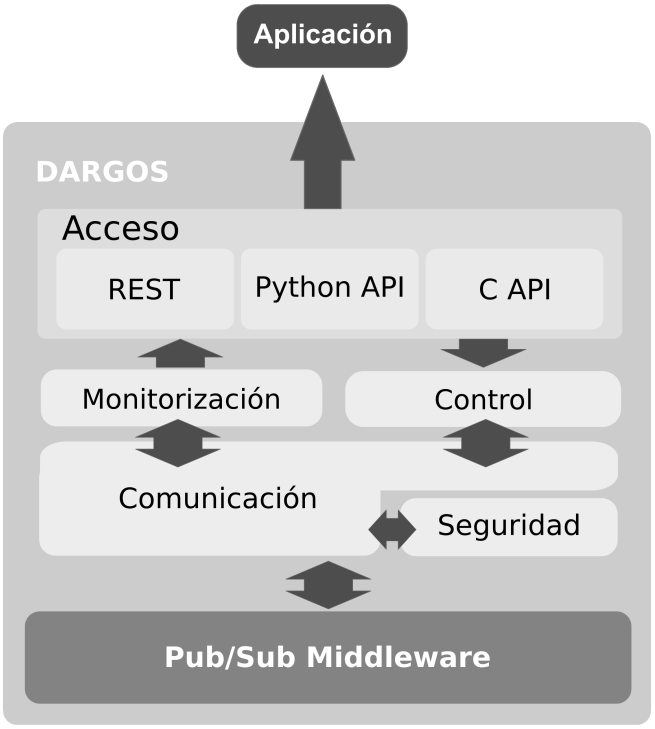


Figura 4.3: Módulos DARGOS.

como *Lattice* basan su comunicación en publicar cada cierto tiempo los datos de monitorización. En ambos casos, dichos datos se publican utilizando el formato de representación XDR en una determinada dirección *multicast*.

Ambos sistemas publican los datos XDR en bruto, sin encapsular en ningún protocolo de control. La única diferencia entre ellos radica en la difusión de la información de formato para cada una de las métricas usadas (metadatos): mientras *Ganglia*, para reducir el tráfico de red, publica los metadatos periódicamente *offline*, *Lattice* los incluye dentro de la información de monitorización con cada muestra *online*.

Ambas soluciones, tanto *Lattice* como *Ganglia*, al limitarse a publicar los datos de monitorización sin ningún protocolo de control asociado, presentan algunas dificultades en algunos escenarios.

En primer lugar, al no existir un protocolo de control no se permite la implementación de entrega de datos fiable, siendo por tanto vulnerables a pérdidas de paquetes.

En segundo lugar, el soporte de filtrado de datos es limitado ya que ante la ausencia de mecanismos que los habiliten, el filtrado sólo se puede realizar en el destino a nivel de aplicación.

Como solución a todas las limitaciones anteriormente expuestas es necesario un sistema de comunicación que soporte los requerimientos deseables descritos en esta sección. El sistema propuesto adopta una aproximación centrada en datos (*data-centric*) para la distribución de datos de monitorización.

Mediante la aproximación *data-centric* la información fluye entre nodos en base al contenido de la misma sin tener en cuenta la localización de la fuente o destinataria de la misma, con ello se logra un nivel extra de desacoplamiento entre emisor y receptor. Esto permite realizar operaciones avanzadas (*e.g.* filtrado) sobre la información de monitorización que no podían ser realizadas por otras soluciones basadas en la mera distribución *multicast* de las mediciones de uso de los recursos.

Dentro de las posibilidades existentes que permitan llevar a cabo una aproximación *data-centric*, es deseable elegir una alternativa que se encuentre estandarizada (requisito identificado en la Sección 4.1). De este modo, se consiguen las siguientes ventajas:

1. Se facilita la integración con sistemas y entornos ya implantados o de futura implantación.
2. Se evitan los costes asociados que pudieran surgir debido a un cambio de proveedor de la tecnología (*vendor lock-in*).
3. La base de usuarios es mayor, por lo que soluciones basadas en estándares suelen ser más estables y maduras.

A tenor de lo anteriormente expuesto, para el intercambio de información de monitorización nuestro sistema propone el uso del estándar de la OMG de publicación/subscripción DDS.

En DDS, la unidad de información mínima intercambiada entre pares es el tópico (*Topic*). Un tópico queda definido mediante un nombre y la definición del tipo de dato que lo compone. Opcionalmente, cada tópico se puede identificar unívocamente mediante una clave (*e.g.* ID). Para más información ver la Sección 2.1.4 del Capítulo 2.

La característica más importante de la aproximación *data-centric* es que el *middleware* puede identificar cada átomo de información y su contenido determina la gestión y difusión del mismo. Este comportamiento difiere de otros sistemas de publicación/subscripción no centrados en datos, ya que en estos casos, el contenido de los mensajes intercambiados es opaco al *middleware* empleado, no permitiendo así condicionar el comportamiento al contenido de los mismos.

Otra característica relevante del estándar DDS es que los requerimientos de difusión y/o opciones de filtrado se pueden definir de manera individualizada para

cada tópico publicado en el sistema. Esta característica de personalización es especialmente relevante para los sistema de monitorización, ya que unos recursos requerirán por ejemplo actualizaciones frecuentes, mientras que otros recursos no.

Así por ejemplo, mientras que el uso de CPU puede variar muy rápidamente, el uso de memoria física es más lento y gradual, por lo tanto publicar actualizaciones muy frecuentes de estado de ciertos recursos aparte de innecesario puede generar excesivo tráfico de red, pudiendo resultar –por lo tanto– contraproducente para el desempeño del sistema.

Después de justificar la elección de DDS, resta definir cómo el diseño propuesto hará uso del *middleware* de publicación/subscripción en el sistema que nos ocupa.

El primer punto a definir es cómo mapear las mediciones recolectadas por los sensores a las unidades de intercambio en DDS.

Algunos sistemas como [116], consideran –por defecto– cada parámetro medible de un recurso como una medida independiente. Aplicar esta aproximación a una solución *data centric* puede acarrear un uso excesivo de recursos, ya que cada uno de estos parámetros tiene asociada metainformación que debe ser transmitida a los nodos interesados y almacenada en una base de datos local, haciendo que en escenarios con muchas métricas el volumen de datos a mantener sea grande.

Por otro lado, existe la alternativa de que cada nodo publique conjunta y periódicamente un resumen del valor de todos y cada uno de los parámetros recolectados. Sin embargo, esta opción conlleva que un nodo pueda recibir mucha más información de la que necesita, lo que puede implicar un uso ineficiente de recursos en determinados casos.

Como alternativa, este trabajo propone una solución híbrida: todas las métricas y parámetros correspondientes a un determinado recurso se publican bajo un único tópico, pero además se publicarán tópicos que consisten en la agregación de mediciones de parámetros de múltiples recursos. La ventaja de dicha aproximación radica en que puede satisfacer tanto a los nodos interesados en monitorizar sólo ciertos recursos, como aquellos que están interesados en tener una fotografía global del estado del sistema (*i.e.* interesados en el estado de todos los recursos) sin necesidad de mantener múltiples subscripciones simultáneas, ahorrando así recursos.

Como se ha mencionado anteriormente, la aproximación *data-centric* empleada por DDS, permite definir tópicos que pueden ser identificados unívocamente mediante una clave única. El sistema presentado utiliza dicha característica para identificar unívocamente los recursos monitorizados.

Concretamente, a cada tópico asociado a un sensor se le asocia un identificador único que es generado de acuerdo con el estándar UUID [107]. UUID está compuesto por 16 octetos (128 bits), por lo tanto el número de identificadores únicos que

Política QoS	Descripción	Aplicación
Liveliness	Mecanismo de control que permite identificar cuando una entidad en el otro extremo se desconecta	Identificar cuándo un nodo monitorizado ha dejado de estar disponible.
Partition	Mecanismo que permite añadir etiquetas a publicaciones subscripciones para realizar un emparejamiento	Dividir conjuntos de nodos en subconjuntos con características comunes.
Reliability	Indica si las muestras perdidas por la red deben de ser reenviadas por el *middleware*	Asegurar que información crítica llega a todos los subscriptores (*e.g.* alarmas).
History	Número de muestras que deben permanecer almacenadas en la cache	Mantener un registro de los últimos valores medidos en un determinado recurso para poder detectar tendencias o predecir comportamientos futuros.
Time Based Filter	Tiempo mínimo entre muestras a recibir	Permite limitar la frecuencia con la que un subscriptor recibe muestras para adaptarse a sus necesidades.

Tabla 4.1: Políticas de calidad de servicio (QoS) aplicables a la monitorización de recursos en entornos distribuidos.

se pueden generar es de $2^{16} \approx 3{,}4x10^3 8$, un número que implica que la probabilidad de colisión sea muy reducida. En el caso de *DARGOS*, se emplea la versión 4 de UUID que constan de 122 bits generados aleatoriamente. De acuerdo a la *paradoja del cumpleaños*[119] la posibilidad de generar un UUID duplicado puede aproximadamente por la Ecuación 4.1, siendo n el número de *UUID* generados y por tanto $p(n)$, la probabilidad de colisión entre n UUID generados aleatoriamente. Como se puede observar, dicha probabilidad $p(n)$, es muy pequeña al ser $\frac{n^2}{22^122}$ un número muy pequeño.

$$p(n) \approx 1 - e^{-\frac{n^2}{22^122}} \tag{4.1}$$

El estándar DDS proporciona además una característica que lo diferencia de

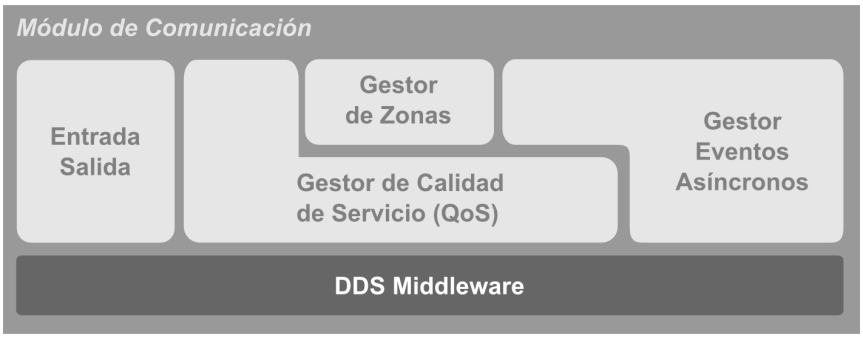


Figura 4.4: Módulo de Comunicación.

otros entornos de publicación/subscripción: DDS permite aplicar calidades de servicio (QoS) a cada tópico de forma independiente y singularizada. Esto permite tener un mayor control sobre cómo se llevará a cabo la difusión de la información de monitorización para cada dimensión de interés. De todo el abanico de QoS especificadas en el estándar DDS, se han seleccionado aquellas de mayor utilidad para el servicio distribuido de monitorización de recursos (Tabla 4.1).

Una vez justificado el uso de DDS como núcleo de comunicación del sistema de monitorización propuesto en esta Tesis, a continuación se explican algunas funcionalidades del módulo de comunicación (mostrado en Figura 4.4) que son posibles gracias a la elección del paradigma de comunicación *data-centric* proporcionado por el *middleware* DDS.

Gestión de zonas. Un sistema distribuido no siempre es homogéneo, por ello no siempre es necesario subscribirse a la información de monitorización de todos y cada uno de los nodos presentes en el sistema. Por lo tanto, es necesario habilitar mecanismos que permitan monitorizar nodos organizados en agrupaciones lógicas de acuerdo a algún criterio establecido por el administrador del sistema.

Así por ejemplo, si en un *data-center* se deseara monitorizar únicamente los nodos que disponen de algún hardware específico –como por ejemplo un *codec* de vídeo hardware–

resultaría ineficiente subscribirse a la información de monitorización de todos los *hosts* disponibles en el sistema para después filtrarla en el destino. Como solución a este aspecto, el sistema presentado propone la organización lógica de nodos en agrupamientos virtuales llamados zonas. De este modo, un nodo supervisor interesado en información de monitorización de un subconjunto de nodos se subscribirá únicamente a los sensores que se encuentran en esa determinada zona. En la

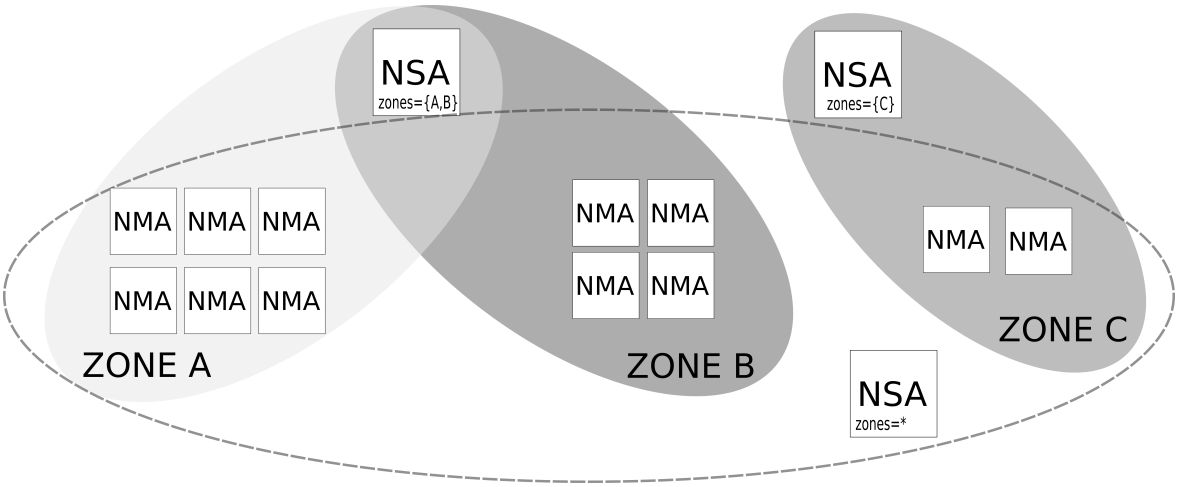


Figura 4.5: Gestión de zonas.

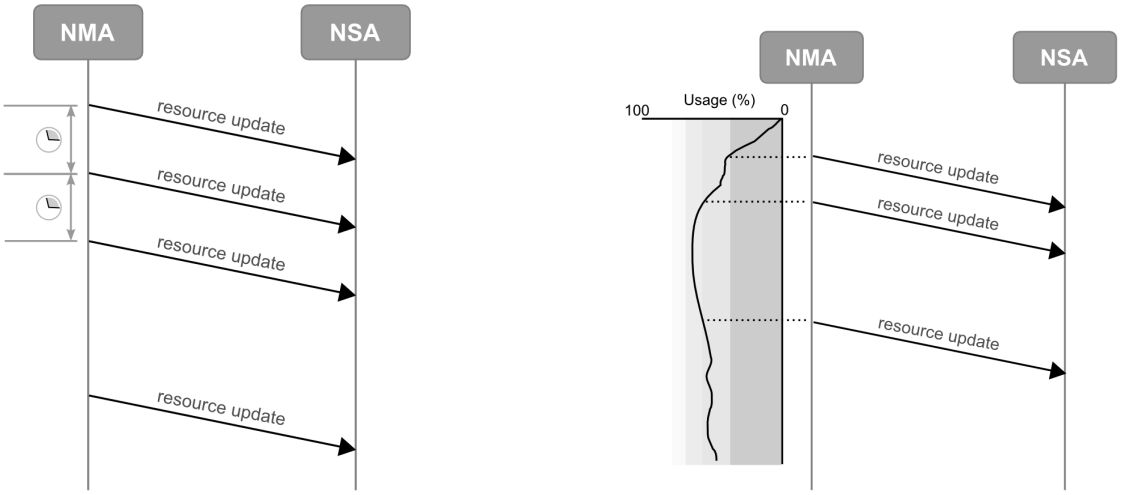
Figura 4.5 se muestra un ejemplo de dicha organización.

Escalabilidad. Cuando el número de nodos monitorizados en un sistema crece, publicar periódicamente el estado de los sensores en los mismos puede resultar costoso en términos de consumo de recursos computacionales y uso de la red, especialmente si los períodos de actualización son pequeños. Para evitar esta situación es deseable habilitar mecanismos que reduzcan el tráfico de monitorización, especialmente cuando las interfaces de red habilitadas estén compartidas con otros protocolos (*e.g.* ssh).

En primera aproximación, para lograr este objetivo se pueden agrupar los *hosts* en zonas y se pueden realizar subscripciones selectivas a determinadas zonas. A pesar de ello, si las zonas son demasiado grandes este problema puede persistir.

Como solución, el sistema propuesto dispone de un modo adicional de monitorización además del periódico. Este nuevo modo, llamado actualización basada en eventos (*event-based update*) consiste en la división del uso de un recurso en rangos de porcentaje. Para ello, se calcula periódicamente el uso de un determinado recurso, pero sólo se publicará dicha información si la última medición ha cambiado de rango desde la última publicación realizada. De esta forma se logra reducir drásticamente el consumo de red, especialmente cuando el sistema se encuentra estable en términos de carga del mismo.

Un hecho relevante en la implementación de este modo es la propuesta de una división no uniforme de los rangos (Figura 4.6). Esta decisión se justifica porque es más importante notificar un recurso cuando está cercano a la sobrecarga, que cuando está en niveles normales o de baja carga. Así por ejemplo, no es tan importante conocer si el consumo de CPU está al 15% o 30%, que si se encuentra en niveles



(a) Periódica (b) Basada en eventos

Figura 4.6: Políticas de actualización: Notificación periódica *vs.* basada en eventos.

entre 75 % u 80 %, siendo en este último caso de mayor importancia llevar un seguimiento del uso de los recursos. El algoritmo de decisión implementado se muestra en Algoritmo 1.

Algorithm 1 Algoritmo de notificación *event-based update*.

```
Require: modoPublicacion
  res = Obtener medición de uso de recurso
  if modoPublicacion = EVENT_BASED then
    rangoActual = Calcular rango actual de uso del recurso
    if rangoActual != rangoUltimoPublicado then
      Publicar res
      rangoUltimoPublicado = rangoActual
    end if
  else
    Publicar res
  end if
```

Eventos asíncronos.

Notificación de alertas Por último, cabe destacar que DARGOS incluye una característica ausente en la mayoría de los demás sistemas, que exprime al máximo el modelo publicación/subscripción: la notificación asíncrona de alertas. La mayoría de los sistemas de monitorización (*Ganglia*, *Lattice* y otros), se limitan a recolectar

estadísticas de uso de recursos y difundirlas de acuerdo a una serie de reglas definidas de antemano. Sin embargo, el sistema aquí presentado proporciona además un mecanismo de alertas asíncronas no presente en otros sistemas, de modo que las eventualidades como fallos o uso excesivos de recursos son publicados como tópicos independientes.

Las alertas normalmente transportan información de mayor prioridad que las actualizaciones de información, por lo que DARGOS considera que deben de enviarse utilizando distintos canales de información. Concretamente la propuesta de DARGOS es la de usar un tópico con unos requerimientos de entrega más exigentes (*e.g.* RELIABILITY) y que además permite la instalación de actuadores que son disparados ante la recepción de alarmas. Estos actuadores permiten realizar acciones orientadas a corregir o mitigar el fallo causado por la alarma, tales como reiniciar un servicio, iniciar nuevos nodos en una red, etc.

Gestor de calidad de servicio Una característica importante del *middleware* de comunicación empleado, es el soporte que proporciona de calidad de servicio. Gracias a esto se puede configurar las calidades de servicio que atañen a cada ítem publicado por DARGOS, que pueden ser estadísticas de uso, alarmas o comandos. Este módulo es encargado de configurar las políticas de calidad de servicio para cada uno de estos ítem o tópicos. Así por ejemplo, como se ha mencionado anteriormente, las alertas requieren de fiabilidad (*RELIABILITY*) en la entrega mientras que las estadísticas de uso dependen de los requerimientos de un usuario determinado. Este módulo es encargado de configurar las calidades de servicio de DDS para cada uno de los tópicos utilizados por DARGOS

4.2.3.2. Módulo de recolección de datos

El núcleo de un sistema de monitorización es el módulo encargado de recolección de métricas y estadísticas del uso de recursos (ver Figura 4.7). Éste es el encargado de recopilar mediante las interfaces dispuestas por el sistema operativo las métricas asociadas a recursos software y hardware (*e.g.* el sistema operativo GNU/Linux proporciona en el directorio */proc* información de uso de algunos recursos como el uso de CPU y memoria).

El tipo de información recolectada por un sistema de monitorización condiciona el tipo de entornos en los que puede ser usado e integrado. En el caso de entornos distribuidos con contenido multimedia, muy demandante en recursos físicos, es interesante tener un control de grano fino sobre el uso que se hace de dichos recursos. De acuerdo a esta premisa, estos son los sensores elementales que incluye por defecto el sistema de monitorización aquí presentado:

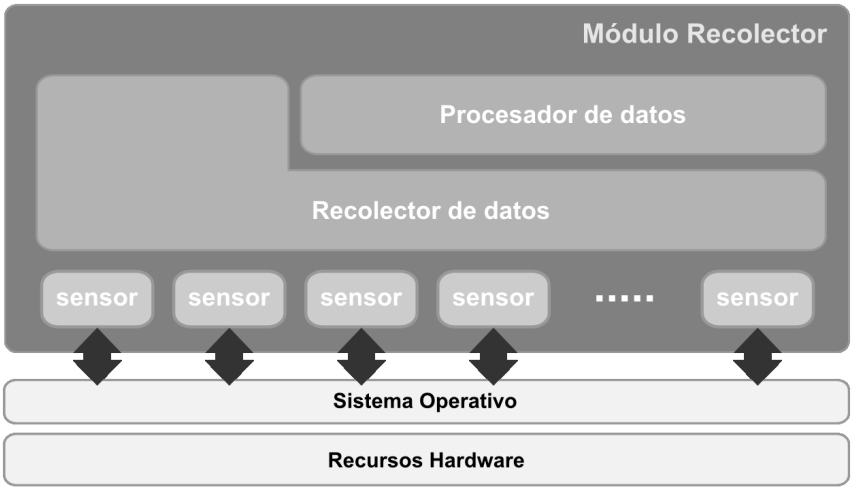


Figura 4.7: Módulo de recolección de datos.

CPU. Medida del porcentaje de uso de CPU. La medición se realiza teniendo el cuenta el tiempo de CPU empleado en un intervalo de tiempo predeterminado. Este sensor permite detectar sobrecargas debidas a aplicaciones muy demandantes de CPU [3].

Memoria. Este sensor proporciona información acerca de la cantidad de memoria física y virtual disponible en el sistema.

Carga del sistema. Este sensor proporciona una medida de la carga de procesos del sistema operativo. Concretamente se toma una medición del número medio de procesos activos en el sistema en periodos de uno, cinco y quince minutos [57].

Interfaces de red. Este sensor lee las estadísticas de uso de cada una de las interfaces de red disponibles en el sistema, pudiendo detectar circunstancias en las que una determinada aplicación esté realizando un uso desmesurado de recursos de red, permitiendo así realizar operaciones de balanceo de carga.

No obstante, el sistema diseñado es totalmente extensible

y permite añadir nuevos sensores que permitan recolectar diversas métricas de nuevos recursos, ya sean software o hardware. Para la definición de un nuevo sensor de recursos basta definir y registrar en el sistema el tipo y formato de datos que proporciona dicho recurso e implementar la recolección de dichos datos mediante llamadas al sistema o el uso de APIs de instrumentación en el caso de servicios y/o aplicaciones.

Hay que tener en cuenta que no todos los recursos hardware pueden ser monitorizados con exactitud usando las métricas primitivas que proporcionan las interfaces del sistema operativo, sino que deben ser derivadas por medio de alguna expresión matemática. Un claro ejemplo de esto es la CPU. Normalmente cuando se habla de uso de CPU se refiere al porcentaje de carga de la misma. En el caso de sistemas *UNIX*, dicha medida no está disponible directamente, sino que habitualmente lo que se proporciona es el número de ciclos (*jiffies*[1]) que han sido consumidos por la CPU. Para conseguir el porcentaje de uso de la CPU, se debe aplicar la expresión 4.2:

$$CPU_t\% = \frac{jiffie_t - jiffie_{t-1}}{\Delta t} \tag{4.2}$$

Debido a esto, el sistema de monitorización debe permitir almacenar un histórico de uso de ciertos recursos para poder así obtener los valores derivados.

4.2.3.3. Módulo de control

Típicamente, los agentes propuestos por otros sistemas de monitorización son agentes pasivos que se limitan a publicar información previamente recolectada. Estos agentes, una vez que son iniciados, requieren de intervención manual por parte del administrador del sistema para modificar alguno de sus parámetros o comportamiento [64].

En algunos escenarios los requisitos, el entorno y condiciones varían a lo largo del tiempo, por lo que una determinada configuración de un sistema de monitorización que puede ser adecuada *a priori*, puede no serlo tanto más adelante. Por ejemplo, en un momento dado puede ser necesario añadir *en caliente* nuevos sensores o por ejemplo modificar la frecuencia de muestreo de los ya existentes para adaptarse nuevas necesidades. Para satisfacer este requerimiento, DARGOS propone el uso de un tópico especial dedicado al envío de comandos a los nodos monitorizados, permitiendo así una mejor gestión de los mismos.

Por medio de los tópicos de comando se posibilita –por tanto– la invocación de comandos remotos en nodos específicos del sistema distribuido. Para ello, el diseño de dicho tópico permite invocar las acciones que sean necesarias para la monitorización en un entorno cambiante. Ejemplos de uso de esta funcionalidad sería por ejemplo el cambio de zona de un NMA, la activación/desactivación de un sensor en un NMA o el incremento o disminución de la tasa de recolecciónd e estadísticas.

Concretamente, el tópico de comando contiene los siguientes campos:

source-id. Identificador del nodo que inicia el comando. Corresponde con el UUID

[1] Un *jiffie* equivale a un 1/10 de segundo

asociado a ese nodo.

destination-id. Nodo al que va dirigido el comando.

command. Comando que se desea ejecutar en el nodo remoto. Por ejemplo, cambio de zona de monitorización, cambio de tasa de recolección de datos, etc.

arguments. Lista de argumentos pasados al comando remoto. Por ejemplo, en el caso de cambio de tasa de recolección de datos, la nueva tasa a emplear.

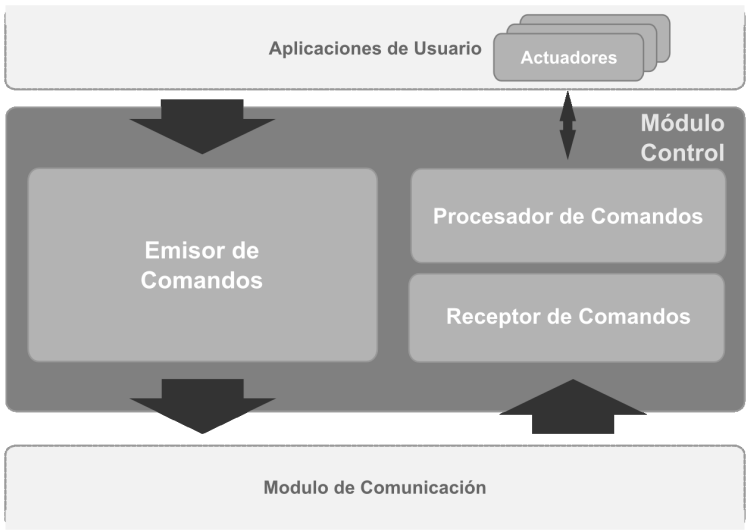


Figura 4.8: Módulo de control.

La ejecución de comandos remotos es una operación delicada que puede comprometer –e incluso acarrear– la inestabilidad del sistema, por lo que es necesario habilitar mecanismos que eviten el envío de comandos por parte de nodos malintencionados. Para ello se debe asegurar que los comandos enviados cumplan unos requisitos mínimos de seguridad tales como la autenticación del origen y la integridad del comando. Para ello se aplicarán los mecanismos de seguridad definidos en la sección 4.2.3.5.

4.2.3.4. Módulo de consulta

Aparte de la capacidad de recolectar y publicar información relacionada con el uso de los recursos, un sistema de monitorización debe proporcionar mecanismos para acceder a la información recolectada desde múltiples fuentes. Éste es el cometido del módulo de consulta (ver Figura 4.9).

El módulo de consulta proporciona las interfaces necesarias para acceder a la información de monitorización que está siendo publicada en un determinado entorno, por lo que este módulo se encuentra situado en el subscriptor.

En el caso de *Ganglia* [116], el acceso a los datos recolectados por un demonio *gmetad* de todo un *data center* se realiza por defecto por medio de un volcado de toda la información almacenada en la cache en formato XML, al que se accede por medio de una conexión TCP. Este tipo de conexiones son más adecuadas para entornos Wide Area Network (WAN) ya que tienen un mejor soporte por parte de dispositivos Network Address Translation (NAT).

En el caso de DARGOS, se utiliza una aproximación similar aunque con algunas variaciones. Concretamente, se han habilitado dos mecanismos distintos para favorecer la flexibilidad de la arquitectura: una API nativa y un servicio REpresentational State Transfer (REST) [99] para el acceso mediante peticiones (*pull*) a la información recolectada por un NSA. De este modo DARGOS proporciona tanto recolección mediante *push* o *pull* en función de los requerimientos de un usuario determinado.

El primer caso consiste en exponer una API nativa que accede a la información almacenada en la memoria cache de monitorización, y que retorna los datos en estructuras nativas de un lenguaje de programación. De este modo, el tratamiento y procesamiento de los datos es más directo, al no tener que realizar peticiones TCP ni transformaciones entre formatos de datos.

Más concretamente se ha expuesto una API para *ANSI C* y otra para el lenguaje *Python*. La elección de estos lenguajes se basa en criterios de eficiencia y flexibilidad: mientras que con C se consigue una mayor eficiencia debido a que el núcleo del sistema está desarrollado en lenguaje C, en el caso de *Python* se ha elegido para facilitar la integración de DARGOS con *scripts* y lenguajes interpretados, lo que mejora su portabilidad y sencillez. No obstante, es posible –dado el diseño modular de DARGOS– exponer otras APIs que sean necesarias, tales como Java.

El mecanismo basado en API nativa es el más adecuado en términos de eficiencia cuando se quiere acceder a la información de monitorización desde dentro de un *data center*, sin embargo esto no es siempre posible. En el código del Listado 4.1 se muestran las operaciones más relevantes de la API nativa de DARGOS. Como se puede observar, se ha optado por mantener una API simple para facilitar el uso e integración con otros sistemas.

La mayoría de los sistemas de monitorización, incluido el que nos ocupa, basan gran parte de su comunicación en el uso de *multicast*, sin embargo este tipo de comunicaciones no siempre se pueden desplegar fuera del entorno del *data center*.

Para solventar este inconveniente, se utiliza la interfaz REST proporcionada por DARGOS. Esta interfaz permite acceder a la información desde fuera de un *data*

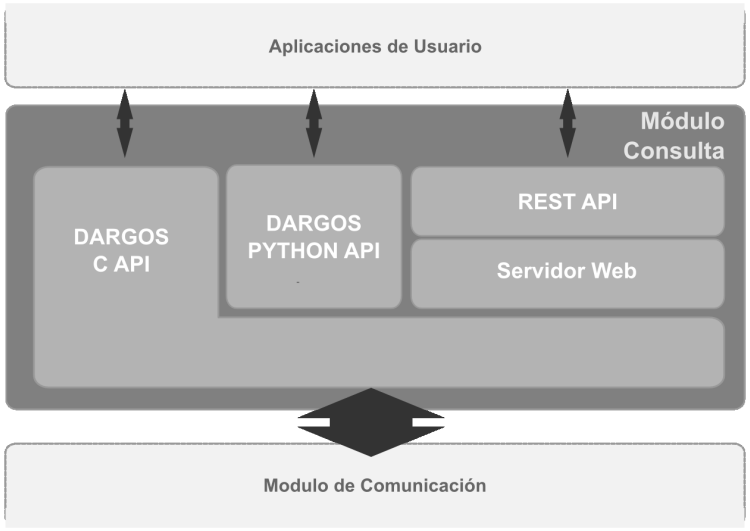


Figura 4.9: Módulo de consulta.

center mediante peticiones HyperText Transfer Protocol (HTTP). Esta aproximación al estar basada en conexiones TCP sobre HTTP evita los problemas relacionados con *multicast* y los derivados del uso de *firewalls* y NAT, si bien esta aproximación sigue adoleciendo de los mismos problemas que se identificaron en cualquier sistema que use TCP (como por ejemplo *Ganglia*).

La aproximación basada en REST presenta una serie de ventajas comparada con otras alternativas similares empleadas en la bibliografía.

En primer lugar, la información se devuelve en formato JSON, una notación mucho más ligera y compacta que XML, lo que reduce considerablemente el tráfico generado y permite una mejor integración con otros servicios web basados en REST.

En segundo lugar, REST utiliza la URL para pasar argumentos de consulta, este hecho simplifica considerablemente el acceso a información de un recurso específico (un determinado sensor, o un determinado *host*). En la Tabla 4.2 se describen las operaciones más relevantes de la API REST diseñada e implementada.

4.2.3.5. Módulo de seguridad

Uno de los aspectos importantes de los sistemas de monitorización que no suele ser tenido en cuenta es el de la seguridad. Los datos deben provenir de fuentes autentificadas y deben estar a salvo de manipulaciones. Ninguno de los sistemas estudiados en el Capítulo 2.3 aborda directamente el tema de la seguridad.

Listado 4.1: DARGOS ANSI C API

```
/* NSA methods */
void nsa_start(NSA, sensor_list, zones);
void nsa_send_command(NSA, node_uuid, command);
status_list nsa_get_sensor_status(NSA, sensor);
status nsa_get_sensor_status_history(NSA, sensor, uuid);
status nsa_set_alert_handler(NSA, alert, handler_function);
void nsa_stop(NSA, sensor_list, zones);

/* NMA methods */
void nma_start(NMA, zone);
void nma_add_sensor(NMA, sensor);
void nma_set_collection_period(NMA, period);
void nma_remove_sensor(NMA, sensor);
void nma_enable_sensor(NMA, sensor);
void nma_disable_sensor(NMA, sensor);
void nma_stop(NMA*);
```

En este sentido, alguno de estos sistemas (*e.g. Nagios* [124]) relegan la seguridad a otros niveles de la arquitectura, ya que utilizan protocolos orientados a conexión como TCP junto con protocolos de aplicación seguros como SSH[194] o TLS [46]. Si bien en sistemas con pocos nodos esta es una aproximación factible, cuando el número de nodos aumenta, los protocolos orientados a conexión no son los más adecuados debido al coste asociado a mantener la conexión, además de por los retardos adicionales incurridos en recuperar errores, en el control de la congestión y para controlar el flujo.

Particularmente -y teniendo en cuenta los objetivos que persigue esta Tesis– se considera que dentro de los requerimientos de seguridad, un sistema de monitorización de recursos distribuido debe verificar los siguientes aspectos:

Autenticación. Los datos deben provenir de una fuente cuya identidad sea fehaciente (es decir, sea quien dice ser). Verificando este aspecto se evita que nodos no autorizados puedan publicar información de monitorización suplantando la identidad de otro nodo.

Integridad. Se debe impedir que los datos puedan ser modificados (voluntaria o involuntariamente) por terceros, de modo que si esto ocurriera, el sistema debe detectar dicha alteración.

En el diseño propuesto e implementación realizada, se adopta la hipótesis de que la confidencialidad de la información de monitorización no es estrictamente

Operación	Descripción
/hosts/	Retorna un resumen del estado de todos los hosts monitorizados.
/hosts/<uuid>	Retorna un resumen del estado del host identificado por el ID uuid
/zones/	Retorna un resumen de las zonas disponibles para monitorizar
/zones/<name>	Devuelve estadísticas de uso de la zona uuid
/cpu/	Retorna el uso de CPU de todos los nodos descubiertos
/cpu/<uuid>	Retorna el uso de CPU del nodo con el ID uuid
/memory/>	Retorna las estadísticas de uso de memoria de todos los nodos descubiertos
/memory/<uuid>	Retorna las estadísticas de uso de memoria del nodo identificado por el ID uuid
/load/	Retorna las estadísticas de carga del sistema de todos los nodos descubiertos
/load/<uuid>	Retorna las estadísticas de carga del sistema del nodo identificado por el ID uuid
/meta/<recurso>	Retorna la metainformación asociada a cada recurso (campos y descripción). Por ejemplo para obtener información acerca del recurso CPU usaría la URL /meta/cpu/

Tabla 4.2: REST API.

necesaria, por lo que dentro de este módulo no se han incluido mecanismos que faciliten este aspecto de la seguridad.

Una justificación adicional para esta decisión es que la no inclusión de cifrado reduce la carga computacional de los nodos. Sin embargo, es posible que en algún escenario la confidencialidad sea un requisito a tener en cuenta. En ese caso se propone utilizar protocolos estandarizados, tales como *IPsec* [98] para nivel de red y DTLS [155] para el nivel de transporte.

Es interesante resaltar que coincidiendo con el periodo de redacción de la Tesis el OMG tiene avanzado la especificación de capacidades de seguridad para el estándar DDS [134]. Ante la imposibilidad de disponer de la versión estable y definitiva de dicha especificación, en la implementación realizada se ha incluido código adicional que proporcione unos servicios de seguridad básicos, explicados más adelante.

Concretamente, en la implementación de DARGOS se ha elegido el estándar HMAC [104, 170]. HMAC permite construir un código de autenticación de mensajes mediante el uso de una función *hash* en conjunción con una clave secreta, lo que permite además asegurar la integridad de los mismos. En la implementación realizada de DARGOS se ha elegido *SHA-1* [50] como función *hash* para firmar (denominada de aquí en adelante *HMAC-SHA-1*[49]).

Para finalizar, remarcar que la inclusión de una firma dentro de un tópico causa un impacto reducido sobre el consumo de red, ya que únicamente supone añadir 160 bits (20 bytes) a cada muestra de tópico. Respecto al impacto en el retardo y uso de recursos por muestra se puede notar igualmente que no es especialmente significativo, debido a que el volumen de información firmado en cada muestra suele ser pequeño (típicamente menor que 8 *kilobytes* [18, 101, 167]).

4.3. Implementación

En esta sección se hará una descripción de los aspectos más relevantes de la implementación de *DARGOS*.

En primer lugar, se ha elegido el lenguaje *ANSI C* para la implementación del núcleo de la infraestructura de DARGOS. Los motivos que han llevado a esta decisión, como se ha mencionado anteriormente, son varios: el primero de ellos es que las prestaciones (*runtime*) de C son mayores que en otros lenguajes interpretados populares como *Java* o *Python* [148], segundo ANSI C es un lenguaje altamente portable, y existen gran cantidad de herramientas para encapsular librerías escritas en otros lenguajes como Python [100] y Java [69].

El núcleo de DARGOS es el encargado de implementar toda su funcionalidad del sistema. Por este motivo, es de vital importancia que se tomen decisiones de implementación orientadas a mejorar la eficiencia del mismo lo máximo posible. Así por ejemplo, para realizar la recolección del uso de los recursos, el sistema de monitorización contiene una hebra dedicada a la recolección, cálculo y almacenamiento de estadísticas. Dicha hebra es independiente de la empleada por el módulo de comunicación, por lo que ante el caso de medidas que no son instantáneas, la publicación del uso de recursos no se verá afectada.

Otro hecho destacable es que DARGOS –como se puede observar en la Figura 4.1– mantiene una jerarquía de clases sencilla y con pocos métodos. El motivo de esta decisión es la de facilitar su uso y facilitar la extensibilidad de la misma. El lenguaje de programación *ANSI C* no soporta programación orientada a objetos, algo necesario para llevar a cabo la implementación de las clases mostradas en la Figura 4.1. Para solventar esto se ha aplicado la metodología expuesta en [164], para

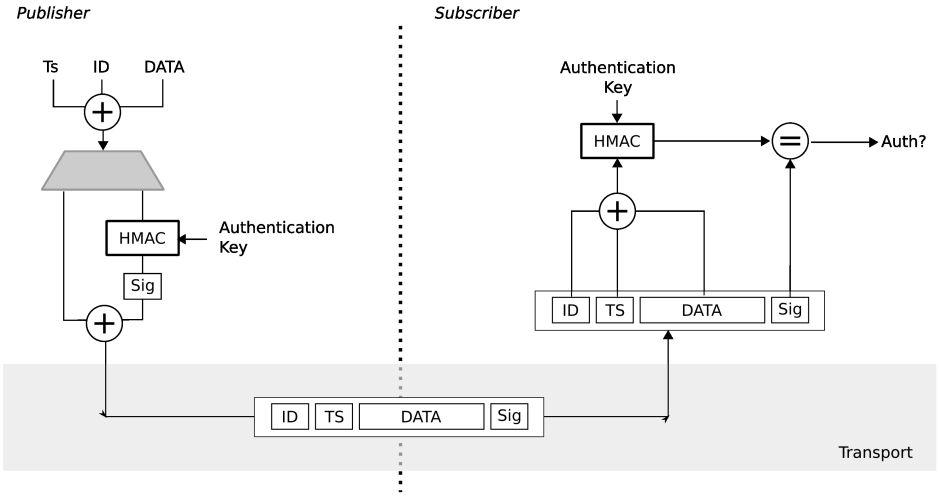


Figura 4.10: Seguridad (autenticación e integridad) en DARGOS.

implementar código orientado a objetos usando *ANSI C*.

Finalmente, aunque el núcleo de DARGOS se ha implementado en ANSI C, se ha optado por proporcionar a DARGOS de una interfaz fácilmente integrable en *scripts*, ya que estos se usan habitualmente en la administración de sistemas. Para ello se ha implementado adicionalmente un recubrimiento ligero escrito en *Python*. Dicho recubrimiento ha sido implementado usando *CTypes*, una librería [100] que permite realizar llamadas nativas al núcleo implementado en C desde *Python*. Con esta aproximación se consigue que la ejecución de código crítico sea responsabilidad del núcleo, mucho más eficiente al estar escrito en C.

4.3.1. Seguridad

Uno de los aspectos relevantes que presenta DARGOS frente a otros sistemas de monitorización es la inclusión de mecanismos de seguridad como parte integral del mismo. Mientras otros sistemas de monitorización estudiados en la Sección 2.3 del Capítulo 2 relegan totalmente esta funcionalidad al protocolo de transporte empleado, DARGOS incluye esta funcionalidad dentro del flujo de tratamiento de información que realiza.

En esta sección se procederá a una descripción del tratamiento de la información llevado a cabo por DARGOS para asegurar las características de seguridad.

El procedimiento es sencillo (ver Figura 4.10): antes de enviar cada mensaje, en el publicador (ver Algoritmo 2) se genera una firma *HMAC* obtenida a partir de la información que se desea publicar: el UUID del origen de los datos, los datos de

Algorithm 2 Algoritmo de generación de firma (publicador).

```
Require: ID, data, Key
  sig = HMAC_Init(Key)
  Ts = time()
  HMAC_Update(sig,ID)
  HMAC_Update(sig,data)
  HMAC_Update(sig,Ts)
  Publish(ID,data,sig,Ts)
```

Algorithm 3 Algoritmo de verificación de firma (suscriptor).

```
Require: Key, timeThreshold, ID, data, Ts
Ensure: authenticated
  (ID,data,originSignature,Ts) = Read_Sample()
  computedSignature = HMAC_Init(Key)
  HMAC_Update(computedSignature,ID)
  HMAC_Update(computedSignature,data)
  HMAC_Update(computedSignature,Ts)
  if originSignature == computedSignature then
    currentTime = time()
    if timediff(currentTime,Ts) ¡timeThreshold then
      return authenticated = True
    else
      return authenticated = False
    end if
  else
    return authenticated = False
  end if
```

monitorización a enviar y la marca de tiempo (a modo de *nonce*) correspondiente al instante en el que se publicaron los datos. Dicha firma una vez generada, se adjunta al mensaje enviado .

En el extremo del receptor (ver el Algoritmo 3 del subscriptor), el proceso es el inverso: se lee la muestra recibida y se genera análogamente la firma correspondiente a la muestra recibida tal y como se hizo en el lado del publicador. Si ambas firmas coinciden, el mensaje se puede considerar como autenticado e integro (siempre que se asuma que la función *hash* es inmune a ataques de colisión). Sin embargo, en el caso de no coincidir se puede concluir que algún usuario malicioso ha intentado suplantar la identidad del publicador DARGOS o que ha habido una alteración (intencionada o no) del mensaje, por lo que en cualquier caso el mensaje debe descartarse.

Hay que tener en cuenta que para evitar ataques de repetición (o *replay*), se incluyen las marcas de tiempo (*nonce*) dentro de la firma. Si ha pasado un umbral de tiempo determinado definido por el administrador desde que se envió el mensaje hasta que se recibe (por defecto un minuto), no se puede asegurar que el mensaje no provenga de de un ataque repetición y por tanto en ese caso debe ser considerado como no valido. Para ello se asume como hipótesis adicional que los relojes de ambas entidades están sincronizados o con un margen pequeño de sincronización.

4.3.2. Implementación de tópicos DDS

Para la implementación de los tópicos, se ha utilizado el estándar *X-Types* de DDS [135]. Este estándar especifica los mecanismos necesarios para soportar nuevos tópicos dinámicamente; es decir, permite descubrir y acceder a tópicos *a priori* desconocidos, sin necesidad de recompilar el código. Otra ventaja de usar esta extensión normalizada es que permite una evolución de los tópicos a lo largo del tiempo, de modo que se pueden añadir nuevos parámetros a cada sensor sin afectar a la compatibilidad con subscriptores ya desplegados.

Además de los datos de monitorización, los nodos DARGOS intercambian otro tipo de información: comandos. Estos comandos son encargados de la gestión remota de nodos. Los comandos normalmente son operaciones punto a punto, hecho que contrasta el paradigma desacoplado de publicación/subscripción de DDS. Para soportar este tipo de operaciones se ha utilizado una funcionalidad presente en la norma DDS: los tópicos filtrados por contenido (*ContentFilteredTopics*). De este modo, se puede hacer que cada nodo suscriptor, solo reciba los mensajes cuyo campo destinatario coincida con el UUID del correspondiente nodo.

#	Nombre	Procesador	RAM	NIC	OS/Software
Testbed-1					
1	Tipo 1	Intel(R) Xeon(R) E5440 @ 2.83GHz	8Gb	1Gbps x 4	Ubuntu 10.04
3	Tipo 2	Intel Core i5 750@2.67GHz	4Gb	1Gbps x 3	Ubuntu 10.04
Testbed-2					
51	Tipo 3	Intel Core i5 750@2.67GHz	4Gb	100Mbps x 4	Ubuntu 12.04

Tabla 4.3: Características hardware/software e identificación de *hosts*.

4.4. Evaluación de DARGOS

Una vez explicado el diseño propuesto y descritos los detalles de la implementación, esta sección incluye la evaluación experimental realizada.

Para ello, en primer lugar se describe el entorno experimental, a continuación en la Sección 4.4.2 se mide el impacto de DARGOS en la cantidad de tráfico generado, concretamente se caracteriza su comportamiento al aumentar el número de publicadores y subscriptores (emulando un *data center* de tamaño medio) y también se evalúa –en términos relativos– comparándolo con otros sistemas de referencia.

En la Sección 4.4.3 se miden retardos incurridos al añadir seguridad y finalmente, en la Sección 4.4.4 se presentan resultados que demuestran la escalabilidad de DARGOS en un despliegue masivo en un entorno WAN real. En este mismo entorno, para un análisis comparativo también se incluyen resultados de otros sistemas de referencia.

Para demostrar las ventajas cuantitativas del sistema propuesto, se ha elegido una metodología basada en la implementación de prototipos y su evaluación posterior en entornos controlados. La justificación de esta elección frente otras alternativas –en particular, frente a evaluar mediante simulación– es la siguiente:

- Una aproximación basada en prototipos permite obtener una estimación más realista (y precisa) del impacto del sistema de monitorización a evaluar.

- Permite la comparación directa con otras herramientas existentes de monitorización, sin necesidad de modelarlas; es decir, sin necesidad de adoptar hipótesis (más o menos contrastables) sobre su diseño, funcionamiento e implementación.

- En la actualidad no existen herramientas de simulación que incluyan al protocolo RTPS, utilizado para el transporte en DDS.

Software	Versión
Wireshark	1.6.7
RTI DDS	4.5d
Lattice	0.6.4
Nagios	3.2
Ganglia	3.1.7

Tabla 4.4: Versiones software empleadas en la experimentación.

4.4.1. Descripción del entorno experimental

Los experimentos se han desplegado en 4 escenarios diferentes. A continuación se describen las particularidades más relevantes de cada uno de ellos.

En la Tabla 4.3 se muestran las características hardware y software y se identifican los equipos utilizados. La Tabla 4.4 detalla las diferentes versiones empleadas en los experimentos.

Sensor	Métrica	Tipo	Descripción
CPU	user	float	Porcentaje de CPU usado por el usuario
	sys	float	Porcentaje de CPU usado por el sistema
	idle	float	Porcentaje de CPU ocioso
Memory	mem_total	integer	Memoria fisica total
	mem_free	integer	Memoria física libre
	swap_total	integer	Memoria de intercambio total
	swap_free	integer	Memoria de intercambio disponible
Load	one	float	Carga media del sistema durante un minuto
	five	float	Carga media del sistema durante cinco minutos
	fifteen	float	Carga media del sistema durante quince minutos

Tabla 4.5: Recursos monitorizados en los experimentos

En la Tabla 4.5 se incluye el formato de cada uno de los sensores utilizados para monitorización en esta experimentación.

Escenario 1 (*Testbed 1*). El primer escenario (mostrado en la Figura 4.11) corresponde a un despliegue sencillo con 4 *hosts* interconectados por un *switch* de 1Gbps. En tres *hosts* del escenario (denominados *Monitored Hosts*) se instalan los agentes recolectores de información local (proveniente de los sensores correspondientes: uso de CPU, memoria y carga del sistema operativo). Las características hardware de estos equipos corresponden al Tipo 2 de acuerdo con la Tabla 4.3.

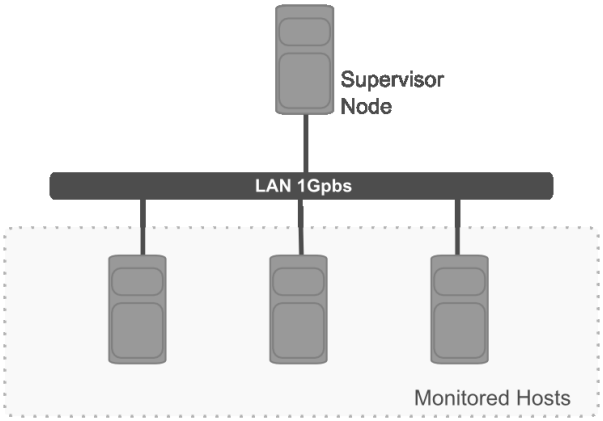


Figura 4.11: *Testbeds* 1 y 2.

Además el escenario contiene un cuarto *host* (*Supervisor Node*, Tipo 1 según la Tabla 4.3) en el que se instala un agente para recolectar toda la información de monitorización.

Escenario 2 (*Testbed 2*). Este escenario (también mostrado en la Figura 4.11) utiliza la misma infraestructura que el *Testbed 1*. La diferencia radica en que en este escenario el número de agentes supervisores en el *Supervisor Node* es variable.

Este escenario es sólo factible para esquemas como *Lattice* y *DARGOS*, ya que sistemas centralizados como *Nagios* no permiten aumentar el número de nodos supervisores en el mismo despliegue.

Escenario 3 (*Testbed 3*). Este escenario (mostrado en la Figura 4.12) evalúa la escalabilidad frente a otras aproximaciones basadas en interacciones de tipo *push*, para ello emula un *data center* de tamaño medio con un mayor número de sistemas monitorizados. Para este despliegue se disponen de 51 *hosts* (50 *Monitored Hosts* más un *Supervisor Node*) interconectados en una Local Area Network (LAN) por medio un *switch* a 100Mbps. En este caso, todos los *hosts* son de Tipo 3 de acuerdo con las características mostradas en la Tabla 4.3.

Escenario 4 (*Testbed 4*). Este último escenario (mostrado en la Figura 4.13) está orientado a la evaluación en escenarios masivos con redes de área amplia (*WAN*). Para ello se monitorizarán 3 *data centers* remotos, cada uno de ellos compuesto por un número distinto de *hosts* (especificados en la Tabla 4.6).

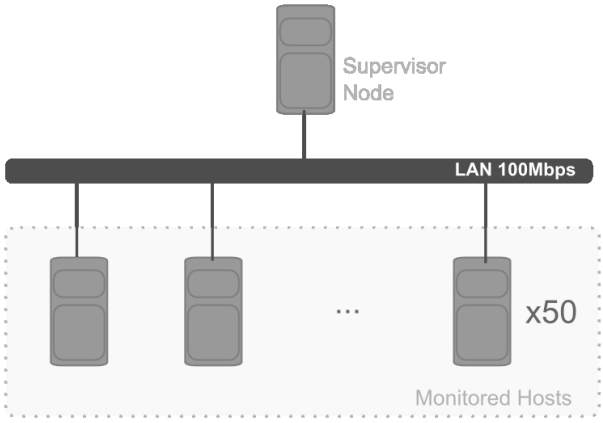


Figura 4.12: *Testbed* 3.

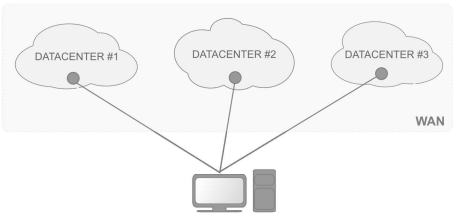


Figura 4.13: *Testbed 4*.

Como se observa en la Tabla 4.6, cada uno de los *data centers* a monitorizar se encuentra situado en una situación geográfica distinta. Concretamente se han considerado 3 localizaciones situadas en 3 Universidades (Bolonia y Nápoles en Italia, y Granada en España), de modo que la experimentación permita simular un entorno lo más cercano posible a un escenario real en el que se ven involucrados varios *data centers* remotos. Esto ha sido posible gracias a la colaboración de grupos de investigación de la Universidad de Nápoles y la Universidad de Bolonia (Italia).

En este escenario, cada *data center* dispone de su correspondiente agente responsable de monitorizar todos las fuentes de monitorización disponibles. Toda la información es recolectada desde un *host* remoto y externo situado en la Universidad de Granada (Tipo 2 según la Tabla 4.3) que se conecta a través de una red *WAN* con cada uno de los *data centers* monitorizados.

Data Center	Location	Hosts
#1 (UNIBO)	Universidad de Bolonia (Italia)	200
#2 (UGR)	Universidad de Granada (España)	100
#3 (UNINA)	Universidad de Nápoles (Italia)	250

Tabla 4.6: Número de *hosts* para el *Testbed* 4.

4.4.2. Evaluación del tráfico generado

Medir el tráfico generado por un sistema de monitorización es importante para determinar su escalabilidad así como su posible coexistencia con otros tráficos o aplicaciones.

La sección se organiza en dos apartados que evalúan distintos aspectos de la arquitectura presentada. En primer lugar, se evalúa la influencia del número de subscriptores y de sensores en el tráfico generado y seguidamente se realizará una comparativa del tráfico generado por DARGOS respecto del tráfico generado por otros sistemas de referencia en las mismas condiciones experimentales.

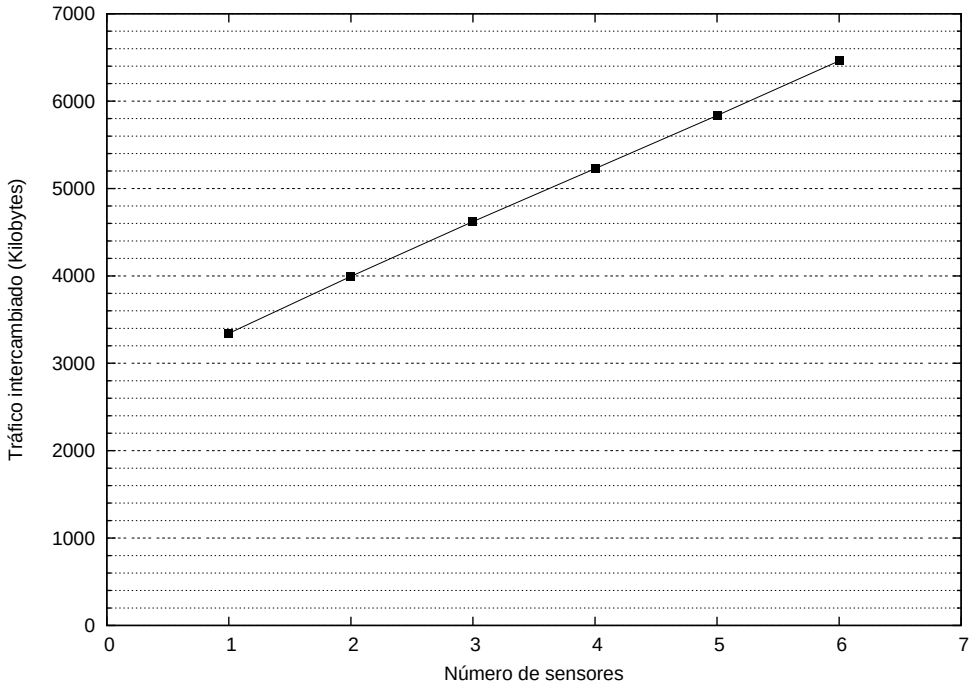


Figura 4.14: Tráfico generado en DARGOS en función del número de sensores.

4.4.2.1. Impacto del número de sensores y subscriptores

Idealmente, un sistema de monitorización debe tener un impacto mínimo en el uso de recursos utilizados. Para evaluar este aspecto de DARGOS se han realizado dos campañas de experimentos.

En primer lugar, se ha medido la influencia del número de sensores en el tráfico generado (Figura 4.14) y en segundo lugar, se evalúa la influencia del número de subscriptores en el sistema (Figura 4.15).

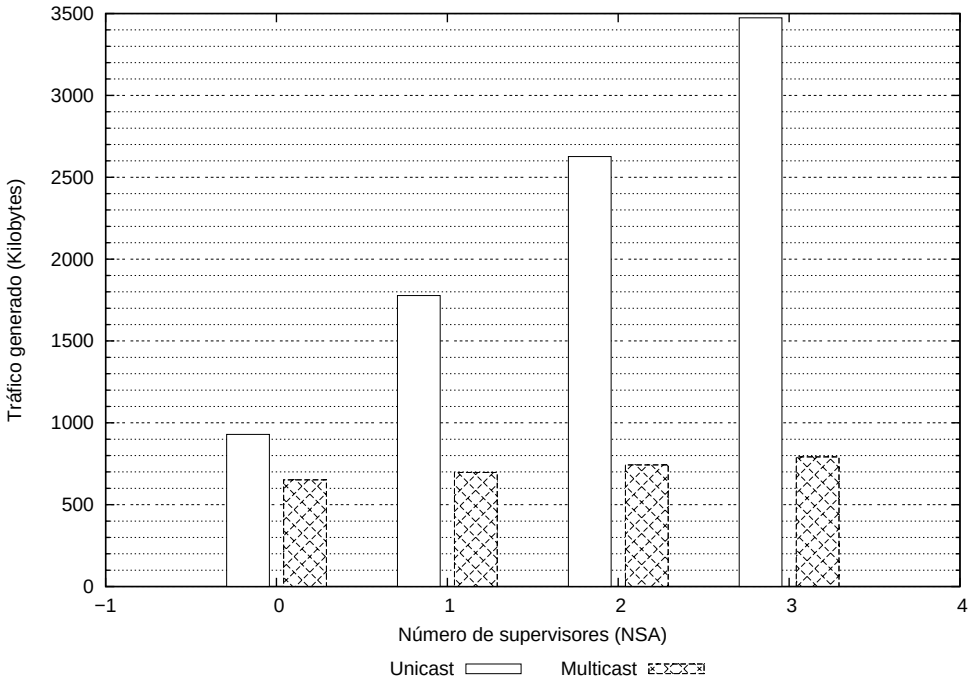


Figura 4.15: Tráfico generado en DARGOS en función del número de subscriptores (NSAs).

Para ello, en el primer caso se han desplegado agentes DARGOS en el *Testbed-1* y se ha medido el tráfico total generado en la red durante un periodo de 300 segundos. Concretamente, se han desplegado hasta 6 agentes NMAs (2 por máquina) con un único sensor habilitado (CPU) repartidos entre los *Monitored hosts* del *Testbed-1*. Cada uno de estos NMA está configurado para utilizar un intervalo de actualización de 1 segundo. El transporte utilizado es UDP *unicast*, ya que al haber un único NSA en este experimento, no influye el uso de *multicast*.

Para el segundo experimento se ha procedido de manera análoga, esta vez utilizando el *Testbed 2*. En este experimento, se ha desplegado un NMAs en cada *host* (cifra constante en todo el experimento), se han ido desplegando sucesivamente NSAs (subscriptores) en cada uno de los *host* disponibles, y se ha medido el tráfico gene-

rado en cada uno de los casos en un intervalo de 300 segundos. Cada uno de los NMA está configurado para publicar una actualización de las estadísticas de uso de CPU por segundo. En este experimento, además se han evaluado dos casos distintos: comunicaciones *unicast* y *multicast*.

Para medir el tráfico en ambos escenarios se ha utilizado el analizador de tráfico *Wireshark* [191]. Concretamente, las medidas corresponden al agregado de todo el tráfico *RTPS* generado por los agentes DARGOS (tanto NSA como NMA) durante la duración de cada uno de los experimentos. Al final de cada ejecución se calcula la suma del tamaño de todos los paquetes RTPS generados en el entorno de red.

Se puede observar claramente que aumentar tanto el número de sensores (Figura 4.14) como el número de subscriptores (NSAs en la Figura 4.15) –con *unicast* y demás condiciones experimentales comentadas– implica un incremento lineal del tráfico generado por DARGOS.

Nótese además, que para comunicaciones *multicast* en el escenario representado en la Figura 4.15, el tráfico generado es prácticamente independiente del número de subscriptores. El ligero aumento observable se debe básicamente a los mensajes *unicast* del protocolo RTPS que por su propia naturaleza no tiene sentido enviarlos en *multicast*. Por ejemplo, los paquetes UDP que contienen confirmaciones (ACKNACK) y los mensajes de mantenimiento de presencia (HEARTBEAT).

4.4.2.2. Comparativa con *Nagios* y *Lattice*

Como sistemas de monitorización de referencia se han elegido a *Nagios* [124] y a *Lattice* [29]. *Nagios* se ha seleccionado por su amplia difusión como herramienta de monitorización de propósito general y por que puede considerarse como un representante significativo de los esquemas de monitorización centralizados.

Lattice, sin embargo, se ha seleccionado por su sencillez y por representar el estado del arte en sistemas de monitorización. Además, puede considerarse como representante de los esquemas de monitorización distribuidos.

En la Tabla 4.7 se incluye una comparación cualitativa entre DARGOS y los sistemas de referencia *Lattice* y *Nagios*. De acuerdo a la misma se puede observar que tanto DARGOS como *Lattice* tienen unas características muy similares en aquellos parámetros relacionados con el modelo de comunicación empleado (transporte de red y cardinalidad). Esto es debido al modelo publicación/subscripción empleado. Sin embargo, existen algunas diferencias importantes entre DARGOS y *Lattice* en cuanto al uso de tecnologías estándares y soporte de operaciones avanzadas sobre datos tales como el soporte de histórico de datos y calidades de servicio.

Se observa además que *Nagios* presenta mayores diferencias con los otros siste-

	DARGOS	Lattice	Nagios
Acoplamiento	Débil	Medio	Fuerte
Soporte QoS	Sí	No	No
Formato de Datos	CDR	XDR	Texto Plano
Datos Históricos	Sí	No	Sí
Notificación	Periódica o Eventos	Periódica, Eventos o Cambio	Depende de *plugin*
Soporte de Filtrado	Basado en tiempo y/o Contenido. En fuente y destino	Basado en Contenido. En fuente	Depende de *plugin*
Transporte	UDP/TCP(REST)	UDP	TCP
Cardinalidad Comunicaciones	N:M	1:N y N:M (sólo Multicast)	1:N
Protocolo normalizado	Si, RTPS	Propietario	Propietario
Notificación Asíncrona	Sí	No	Sí
Señalización Metadatos	*out of band*	*in band*	*in band*

Tabla 4.7: Evaluación cualitativa de DARGOS, *Lattice* y *Nagios*.

mas, fundamentalmente debido a las limitaciones que impone su arquitectura centralizada y al uso del paradigma cliente/servidor.

Para completar la comparación se ha realizado una evaluación cuantitativa. En este caso los experimentos se han realizado en el *Testbed 1* (descrito en la Sección 4.4.1). Para *Nagios* (ver Sección 2.3.3 del Capítulo 2) se evaluarán las dos alternativas que proporciona: NRPE (*pull*) y NSCA (*push*). *Lattice* se evalúa también en *unicast* en modo *push*.

Los sistemas a comparar se han configurado de la manera más equitativa posible. En particular, en todos los casos la tasa de actualización de mediciones se fija a 1 muestra por segundo y en todos los casos se usa el mismo tipo de sensores.

En la primera comparativa, al igual que en experimentos anteriores, se evalúa la influencia del número de publicadores con información de monitorización en el tráfico de red generado para cada uno de los sistemas considerados. Nótese, que al no existir múltiples subscriptores en este experimento, no se han considerado comunicaciones *multicast*, ya que no aportaría ninguna ventaja y obteniéndose resultados similares a *unicast*.

La evaluación realizada consiste en una ejecución de 300 segundos. Respecto a la configuración se han tenido en cuenta 5 casos diferentes:

dargos-unicast. Este experimento corresponde al uso de DARGOS utilizando pro-

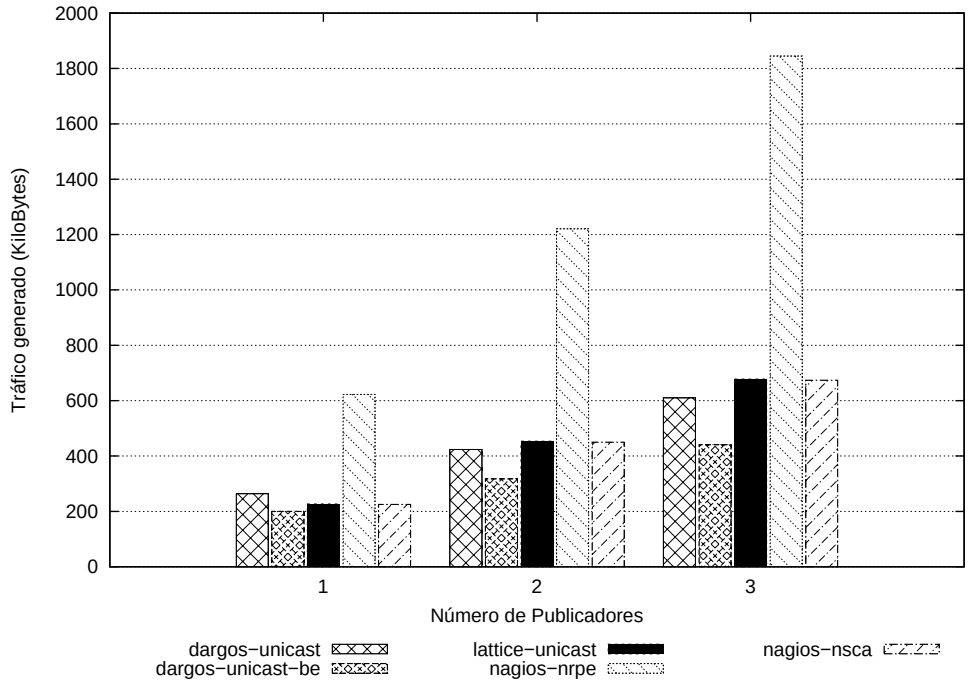


Figura 4.16: Comparativa del impacto del número de publicadores en *DARGOS*, *Lattice* y *Nagios*.

tocolo *RTPS* con fiabilidad activada (QoS `reliability = reliable`, de modo que si ocurren pérdidas o errores el *middleware* los recuperará.

dargos-unicast-be. Este caso consiste en DARGOS utilizando comunicaciones sin fiabilidad (sin corrección de errores) con protocolo RTPS, es decir QoS `reliability = best effort`, de modo que no hay mecanismos de recuperación de muestras (o paquetes) perdidos.

lattice-unicast. Este caso corresponde a un despliegue de monitorización basado en *Lattice* utilizando comunicaciones *unicast* con UDP.

nagios-nrpe. Despliegue basado en el *plugin* NRPE de Nagios. En este caso existe un agente desplegado en cada nodo a monitorizar que recibe peticiones remotas desde el servidor para la ejecución de *scripts*. Estos recolectan la información de monitorización. Este caso iplica una aproximación *pull*.

nagios-nsca. Despliegue basado en el *plugin* NSCA de Nagios. Este *plugin* adopta una aproximación *push*. Para ello se ejecuta una orden desde la línea de comandos que envía la información de monitorización de un determinado recurso al servidor central.

Los resultados obtenidos se muestran en la Figura 4.16. Es resaltable que en

todos los casos *Nagios* genera más tráfico que DARGOS y *Lattice*. Esto hecho se debe fundamentalmente a que las comunicaciones basadas en TCP requieren un ancho de banda mayor, ya que tanto *Lattice* como DARGOS emplean UDP en sus agentes de monitorización. Nótese que no se ha tenido en cuenta la monitorización mediante servicios REST en DARGOS, ya que esta funcionalidad es implementada por los NSAs, no por los NMA.

Otro hecho destacable es que en *nagios-nrpe*, el ancho de banda aumenta considerablemente comparado con otras alternativas. La justificación reside en la naturaleza *pull* de *nagios-nrpe*: por cada actualización de datos de monitorización es necesaria una petición y una respuesta, hecho que no ocurre en aproximaciones tipo *pull*.

Respecto a la comparativa entre *Lattice* y *DARGOS* se pueden destacar los siguientes hechos relevantes.

El tráfico generado por DARGOS se mantiene por debajo de *Lattice* en todos los casos. La justificación reside en las diferencias en el tráfico de metadatos (relativo al formato de cada muestra) de cada sistema: mientras que en *Lattice* cada muestra lleva adjuntos sus metadatos correspondientes (adopta un esquema de señalización *in-band*), DARGOS utiliza una señalización fuera de banda (*out-of-band*), ya que los metadatos se intercambian en la fase de descubrimiento.

Se observa además otro hecho destacable, en el esquema *dargos-best-effort* el ancho de banda es aún menor que en *dargos-unicast*. El motivo de este descenso de tráfico está en la misma naturaleza *best-effort* del tráfico de monitorización, ya que en este caso RTPS no genera el tráfico adicional necesario cuando se usan comunicaciones de tipo *reliable*.

Para analizar el impacto del número de subscriptores sólo se ha considerado a *Lattice* en la comparativa. Esto es debido a que *Nagios* –por su naturaleza centralizada– sólo soporta un único consumidor de información (subscriptor), por lo que su inclusión en esta comparativa carece de sentido.

En éste último experimento se mide el tráfico de red generado al aumentar el número de subscriptores mientras se mantiene constante el número de publicadores/sensores (en este caso 30) desplegados en el sistema. De acuerdo a la Figura 4.17 –en la que se representan los resultados obtenidos– se pueden destacar varios hechos.

En todos los casos DARGOS se comporta mejor que *Lattice*. En comunicaciones *multicast* el tráfico generado se mantiene para ambos esquemas aproximadamente constante, mientras que para el tráfico *unicast* en ambos sistemas se observa una dependencia lineal, como era previsible.

La diferencia entre el tráfico generado por DARGOS y el generado por *Lattice* se

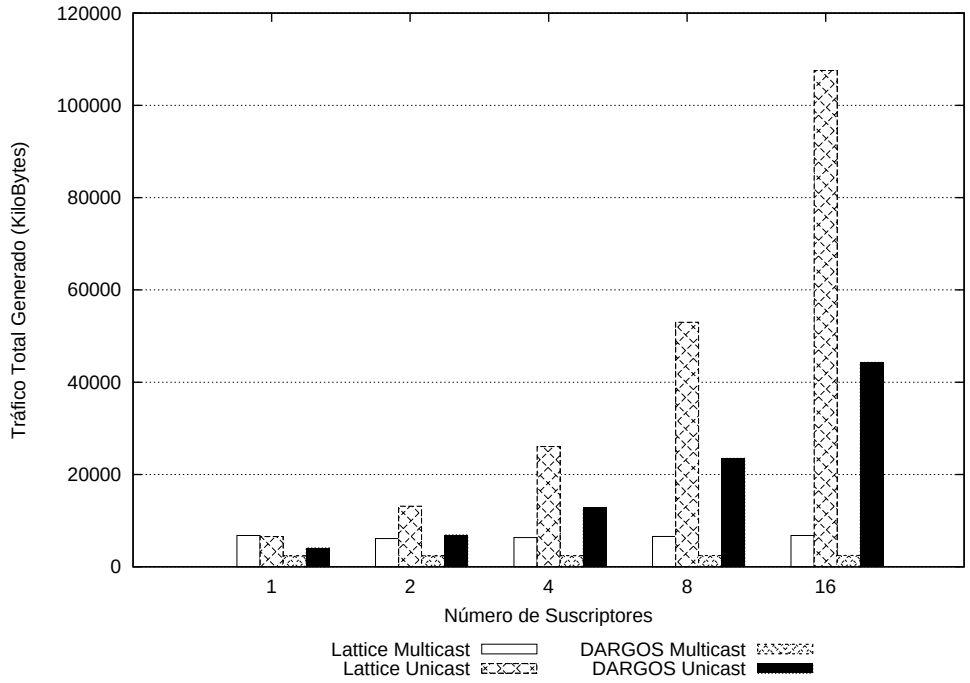


Figura 4.17: Comparativa del impacto del número de subscriptores en DARGOS y *Lattice* para 30 publicadores.

debe a las mismas razones esgrimidas que en el experimento anterior.

4.4.3. Impacto de la seguridad

DARGOS es el único sistema de monitorización –hasta donde alcanza la revisión bibliográfica realizada– que incluye mecanismos de seguridad integralmente dentro de su arquitectura.

Uno de los inconvenientes que puede resultar de esta decisión es penalizar el rendimiento del sistema. En este apartado se evaluará el retardo introducido por la inclusión del esquema de seguridad propuesto (con garantías de integridad y autenticación, ver Sección 4.2.3.5) dentro de la arquitectura de DARGOS. Nótese que en esta evaluación se tiene en cuenta el impacto de incluir los mecanismos de seguridad que DARGOS introduce, no teniendo en cuenta los posibles mecanismos de seguridad externos que podrían introducirse a nivel de middleware DDS en un futuro [134]. No obstante, los mecanismos de seguridad propuestos por DARGOS no son incompatibles con el uso de protocolos de transporte seguros como DTLS.

Para medir el impacto del módulo de seguridad, se realiza un experimento local consistente en medir el tiempo desde que se realiza la medida de un recurso en

un sensor dado hasta que dicha información está disponible en el NSA del nodo supervisor. Nótese que en este caso, ambos agentes se localizan en el mismo nodo por lo que se elimina la influencia de los retardos de propagación y transmisión, además se evita la sincronización de relojes.

Se han considerado por lo tanto dos casos: el primero con el módulo de seguridad habilitado y el segundo sin el. Para cada caso, el experimento se ha repetido 100 veces, obteniendo el valor medio y la varianza.

La Tabla 4.8 muestra los resultados obtenidos para cada una de las configuraciones.

SEGURIDAD	Retardo medio (*msec*)	Varianza
SI	0.303032	0.000056992
NO	0.222843	0.000043451

Tabla 4.8: Impacto del módulo de seguridad.

Se observa que el impacto del módulo de seguridad es relativamente pequeño, alrededor de 80 μsec. Hay que tener en cuenta que este valor incluye una firma en el origen y otra operación de firma para verificar el contenido en el destino.

4.4.4. Escalabilidad del sistema

Otro de los aspectos clave a la hora de validar un sistema de monitorización es su escalabilidad. En algunos escenarios el número de elementos a monitorizar puede crecer notablemente, alcanzando cifras del orden de cientos o miles de *hosts* [13, 67].

Ante esta posible situación, idealmente un sistema de monitorización debe estar diseñado para poder mantener las prestaciones con independencia del tamaño del sistema a monitorizar.

En esta Sección se compara el comportamiento de la arquitectura propuesta (DARGOS) frente a otros sistemas de monitorización relevantes.

Como sistemas de referencia en esta Sección se elige *Lattice* y *Ganglia*. Esta elección se debe a que ambos sistemas fueron concebidos originariamente para monitorizar grandes sistemas (concretamente para entornos de *Cloud Computing* en el caso de *Lattice* [30] o entornos *Grid* en el caso de *Ganglia* [116]), por lo que su diseño tiene en cuenta el problema de la escalabilidad.

Los experimentos de escalabilidad se han realizado usando dos escenarios. En el primer escenario se mide el tráfico generado en la monitorización de un único *data*

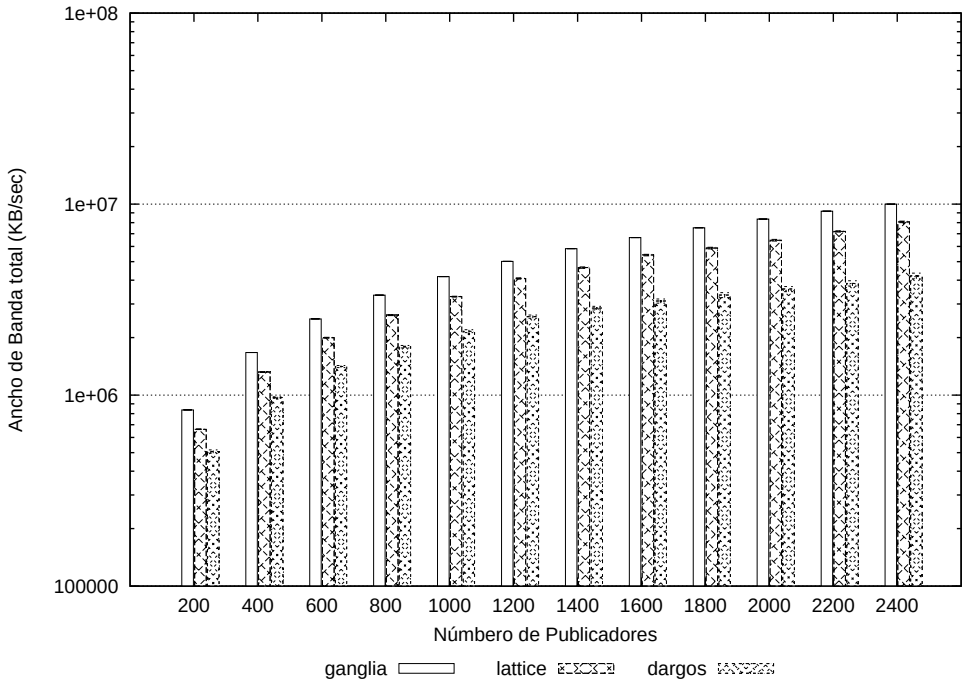


Figura 4.18: Comparación throughput: DARGOS *vs.* Lattice.

center de tamaño medio (correspondiente al *Testbed 3* de la Sección 4.4.1), mientras en el segundo escenario se medirá el comportamiento cuando se monitorizan múltiples *data centers* de mayor tamaño (*Testbed 4*) interconectados por medio de WAN.

Comportamiento en un *data center* de tamaño medio. En este caso, los tres sistemas de monitorización considerados (*DARGOS*, *Lattice* y *Ganglia*) han sido configurados y desplegados con una configuración equivalente: se realizarán actualizaciones periódicas de monitorización utilizando comunicaciones *multicast*. Nótese que para los tres casos se han utilizado los mismos sensores (CPU y memoria)

y que el periodo de actualización es el mismo en los tres casos (0.2 segundos). La duración de cada ejecución ha sido de 2 minutos (120 segundos).

La elección de este elevado periodo de actualización, sitúa al sistema en un punto de operación con una carga superior a las habituales. Con esto se consiguen dos objetivos: evaluar la robustez del sistema y poder predecir el comportamiento en despliegues más grandes con un numero menor de nodos.

Así por ejemplo, la carga generada por 50 *hosts* con una frecuencia de actualización de 5 muestras por segundo, equivaldría a la generada en un sistema de 250 *hosts* con una frecuencia de actualización de 1 muestra por segundo.

N_{Pub}	DARGOS	Lattice	Mejora %	Ganglia	Mejora %
200	506280	662935	23.63	836059.667	39.44
400	971140	1320130	26.44	1670860.33	41.88
600	1411370	1996640	29.31	2504408	43.64
800	1793605	2624110	31.65	3340744.33	46.31
1000	2160955	3277880	34.07	4176080	48.25
1200	2575915	4080240	36.87	5010796	48.59
1400	2854400	4652290	38.65	5844326.33	51.16
1600	3107700	5413800	42.60	6680356.33	53.48
1800	3339885	5885300	43.25	7516159.67	55.56
2000	3591160	6475200	44.54	8351575.33	57.00
2200	3849275	7195500	46.50	9178066.33	58.06
2400	4208610	8067850	47.83	10003047.3	57.93

Tabla 4.9: Tráfico total generado (Bytes/segundo) por DARGOS, Lattice y Ganglia y los respectivos porcentajes de mejora de DARGOS.

Como métrica se ha considerado el *rate* o cantidad de datos por unidad de tiempo. Para la obtención de dicha medida, se han capturado todos los paquetes *multicast* generados en la red debido a la monitorización de los nodos, y en base a ellos se ha calculado la correspondiente tasa.

Se observa en la Figura 4.18 que incluso en escenarios grandes (por debajo de 2500 publicadores), DARGOS obtiene mejores resultados que *Lattice* y *Ganglia*. Más concretamente, la tendencia general es que la pendiente es menor en DARGOS que en los sistemas de referencia, luego se concluye que dentro de un *data center* el esquema propuesto –bajo las suposiciones y condiciones del experimento– escala mejor que los dos sistemas de referencia considerados. En la Tabla 4.9 se muestran los porcentajes de mejora relativa.

Comportamiento en *data centers* en WAN. En algunos escenarios –típicamente en redes que se encuentren detrás de cortafuegos (*firewalls*)– no siempre es posible desplegar aplicaciones basadas en UDP que sean universalmente accesibles. Además, a pesar de que algunos de los sistemas de monitorización descritos en el Capítulo 2.3 basan su operación en mensajes *multicast* (*e.g.*, *Ganglia y Lattice*), es sabido que las comunicaciones *multicast* no están necesariamente habilitadas en todas las redes (especialmente en WAN).

Con estos precedentes, para facilitar la monitorización de sistemas distribuidos en redes WAN es necesario habilitar algún mecanismo que posibilite la compartición de la información de monitorización desde localizaciones remotas de una manera accesible y escalable.

Data Center	min	avg	max	mdev
UNINA	59.123	59.179	59.235	0.328
UGR	0.131	0.143	0.164	0.018
UNIBO	37.826	37.918	38.147	0.274

Tabla 4.10: *Round trip time* entre el recolector de datos y los *data centers*

En este sentido, DARGOS y *Ganglia* son los únicos sistemas que consideran esta problemática. Para ello, proporcionan interfaces que permiten consultar toda la información monitorizada desde localizaciones remotas (puede que en WAN). Estas interfaces agregan la información de monitorización disponible, tal que pueda ser consultada mediante peticiones explícitas (adoptando un modelo *pull*).

Ganglia permite acceder a un volcado de la información monitorizada mediante el establecimiento de una conexión a un puerto determinado (normalmente 8651) gestionado por un *daemon* (concretamente *gmetad*). Al establecer dicha conexión, el *daemon* devuelve en formato XML un volcado del estado de todas las entidades de monitorización.

La aproximación en DARGOS es distinta: nuestro sistema propone asociar un servicio REST [185] a un NSA, permitiendo así acceder a subconjuntos de información de monitorización por medio del acceso a una URL determinada. Así por ejemplo es posible acceder a los datos de monitorización de una determinada zona, de un determinado host, o un tipo de recurso, todo esto dependiendo de la petición REST realizada. A diferencia de *Ganglia* el servicio REST devuelve la información en formato JSON, mucho más simple y compacto que XML [159].

Para cuantificar las prestaciones de cada una de estas aproximaciones, se ha realizado un experimento que mide el tiempo y el tráfico generado en la recolección de datos de monitorización provenientes de tres *data centers* distintos (*Testbed 4*). En particular, se instala un servicio de agregación (*gmetad* en el caso de *Ganglia* y un *NSA/REST* en el caso de *DARGOS*) por cada *data center* monitorizado. Estos servicios de agregación recopilan la información de tantos *hosts* en cada *data center* como se indican en la Tabla 4.6. Para cada *host* se obtendrán estadísticas de uso de CPU, memoria y carga del sistema.

En este experimento, al evaluarse la eficiencia del mecanismo de tipo *pull* que presentan ambos sistemas de monitorización para obtener la información recopilada dentro de un *data center*, es independiente de los periodos de actualización de información elegidos por cada agente desplegado en cada host monitorizado.

El experimento consiste en medir tanto el tiempo empleado en llevar a cabo una recolección de toda la información de monitorización de los tres *data centers* remotos (ver Figura 4.13) desde una localización exterior, como en medir el tráfico generado

Listado 4.2: Formato REST de DARGOS

```
[{
  "last_update": {
    "tv_sec": 1368690713,
    "tv_nsec": 296522999
  },
  "zone": "zone-a",
  "hostname": "prague",
  "id": "e596c6150d1a4be7ab529dbe39e9f01e",
  "cpu": {
    "timestamp": {
      "tv_sec": 1368690714,
      "tv_nsec": 310688999
    },
    "cpu_sys": 2.4752475247524752,
    "cpu_user": 2.9702970297029703,
    "cpu_idle": 53.960396039603964
  }
}]
```

(tamaño de mensajes).

En los Listados 4.2 y 4.3 se muestra un ejemplo de los formatos empleados para el intercambio de datos por DARGOS y *Ganglia*. Por motivos de concisión, se han incluido únicamente datos sobre información de CPU. En el caso de *Ganglia* además se ha omitido además la información de formato (Document Type Definition (DTD)), que es incluida en todas y cada una de las respuestas emitidas por el demonio *gmetad*. Se observa en ellas como *Ganglia* incluye en línea (*inline*) metainformación acerca de cada recurso medido, incrementando significativamente el tamaño del mensaje. Esto no es necesario en DARGOS ya que dicha información se puede solicitar aparte.

Para cada caso, se han realizado secuencialmente 100 peticiones de toda la información de monitorización recolectada por el agente correspondiente (*gmetad* en *Ganglia* y NSA en DARGOS) desde una localización externa a los *data centers* situada dentro de la Universidad de Granada y se han calculado los valores medios de tiempo y de tráfico involucrados. El número de *hosts* monitorizados en cada *data center*, se muestran en la Tabla 4.6. La latencia entre el recolector de datos y los distintos *data centers* medida como el *round trip time* se muestra en la Tabla 4.10.

Los resultados (ver Figura 4.21) pone de manifiesto que para las condiciones experimentales indicadas, DARGOS es más eficiente que *Ganglia* en consumo de

ancho de banda requerido. Esto es ocasionado debido al tamaño de mensaje de cada petición, como puede observarse en las Figuras 4.20 4.19.

Esto es debido a que la aproximación de DARGOS utiliza una notación más compacta que *Ganglia*, requiriendo por tanto menos bytes para transmitir la información. A este hecho hay que añadirle que la aproximación REST de DARGOS permite al cliente seleccionar mediante su API subconjuntos de toda la información capturada, evitando tener que enviar el estado de todo el *data center*. Estos hechos conllevan una reducción del ancho de banda requerido demostrando así su escalabilidad.

4.5. Conclusiones

En este capítulo se propone *DARGOS*, un sistema de monitorización basado en el modelo publicación/subscripción centrada en datos destinado a la monitorización de recursos hardware y software en entornos distribuidos.

DARGOS elimina las carencias encontradas en otros sistemas de monitorización de referencia para entornos altamente dinámicos y con requerimientos de tiempo real.

DARGOS se basa en la aproximación DCPS que proporciona DDS. Su diseño satisface los requerimientos de monitorización identificados para los sistemas de distribución y procesado de datos multimedia entre los que destacamos flexibilidad a la hora de seleccionar y añadir nuevos sensores que se adecúen a las necesidades de la aplicación, mínimo impacto introducido y la escalabilidad del sistema.

A pesar de que DARGOS comparte con otros sistemas de referencia algunos rasgos comunes (*e.g.* aproximación *push*, uso de *multicast*) su diseño incluye aportaciones novedosas y eficientes.

En primer lugar, el sistema propuesto se basa íntegramente en tecnologías estandarizadas: el modelo de datos, el *middleware* utilizado, además de los protocolos de capas inferiores. Esto facilita su integración con sistemas ya existentes o futuros.

Otra novedad importante es que su diseño propone el uso de políticas de calidad de servicio QoS para la distribución de la información de monitorización de recursos. Esto permite tener un mayor control en la distribución de la información, por lo que se mejora la adaptación a las particularidades del escenario de operación concreto, lo que se traduce en una mayor facilidad para identificar problemas en el *data center* monitorizado. Esta funcionalidad es un cambio sustancial respecto a otros sistemas, ya que en su mayoría –como por ejemplo *Ganglia* [116] y *Lattice* [30]– se limitan a ser meros agentes que publican información de manera periódica, sin incluir mecanismos de recuperación de errores ni de adecuación al entorno.

Otro hecho importante es que DARGOS propone la adopción de una aproximación *data centric* para la difusión de información de monitorización de recursos en un sistema distribuido. Esto implica un desacoplo espacial y temporal de los generadores y consumidores de información, así como de las aplicaciones, lo que facilita su despliegue e integración en sistemas distribuidos.

Finalmente, otra aportación importante de DARGOS es que es pionero en considerar como parte integral del mismo mecanismos de autenticación e integridad de la información de monitorización publicada. Esta funcionalidad mejora la seguridad del sistema, reduciendo las posibilidades de suplantaciones de identidad y posibles alteraciones -deseadas o no- del contenido de la información.

A tenor de la experimentación realizada, se demuestra que DARGOS mejora la escalabilidad a la vez que muestra mejoras de hasta el 58 % en el impacto sobre el tráfico de red generado respecto a otros sistemas de monitorización disponibles de la bibliografía. A la vez que proporciona el soporte de calidades de servicio, tales como retardo y garantía de entrega, necesarias para monitorizar entornos distribuidos multimedia.

Listado 4.3: Formato XML de Ganglia

```
<!-- DTD (52 lines ommited) -->
<GANGLIA_XML VERSION="3.1.7" SOURCE="gmond">
<CLUSTER NAME="unspecified" LOCALTIME="1368690922" OWNER="
    unspecified" LATLONG="unspecified" URL="unspecified">
<HOST NAME="prague.ugr.es" IP="150.214.203.174" REPORTED="
    1368690922" TN="0" TMAX="20" DMAX="0" LOCATION="unspecified"
    GMOND_STARTED="1367242550">
<METRIC NAME="cpu_idle" VAL="97.6" TYPE="float" UNITS="%" TN="10"
    TMAX="90" DMAX="0" SLOPE="both">
<EXTRA_DATA>
<EXTRA_ELEMENT NAME="GROUP" VAL="cpu"/>
<EXTRA_ELEMENT NAME="DESC" VAL="Percentage of time that the CPU or
    CPUs were idle and the system did not have an outstanding disk I
    /O request"/>
<EXTRA_ELEMENT NAME="TITLE" VAL="CPU Idle"/>
</EXTRA_DATA>
</METRIC>
<METRIC NAME="cpu_user" VAL="0.7" TYPE="float" UNITS="%" TN="10"
    TMAX="90" DMAX="0" SLOPE="both">
<EXTRA_DATA>
<EXTRA_ELEMENT NAME="GROUP" VAL="cpu"/>
<EXTRA_ELEMENT NAME="DESC" VAL="Percentage of CPU utilization that
    occurred while executing at the user level"/>
<EXTRA_ELEMENT NAME="TITLE" VAL="CPU User"/>
</EXTRA_DATA>
</METRIC>
<METRIC NAME="cpu_system" VAL="0.4" TYPE="float" UNITS="%" TN="10"
    TMAX="90" DMAX="0" SLOPE="both">
<EXTRA_DATA>
<EXTRA_ELEMENT NAME="GROUP" VAL="cpu"/>
<EXTRA_ELEMENT NAME="DESC" VAL="Percentage of CPU utilization that
    occurred while executing at the system level"/>
<EXTRA_ELEMENT NAME="TITLE" VAL="CPU System"/>
</EXTRA_DATA>
</METRIC>
</HOST>
</CLUSTER>
</GANGLIA_XML>
```

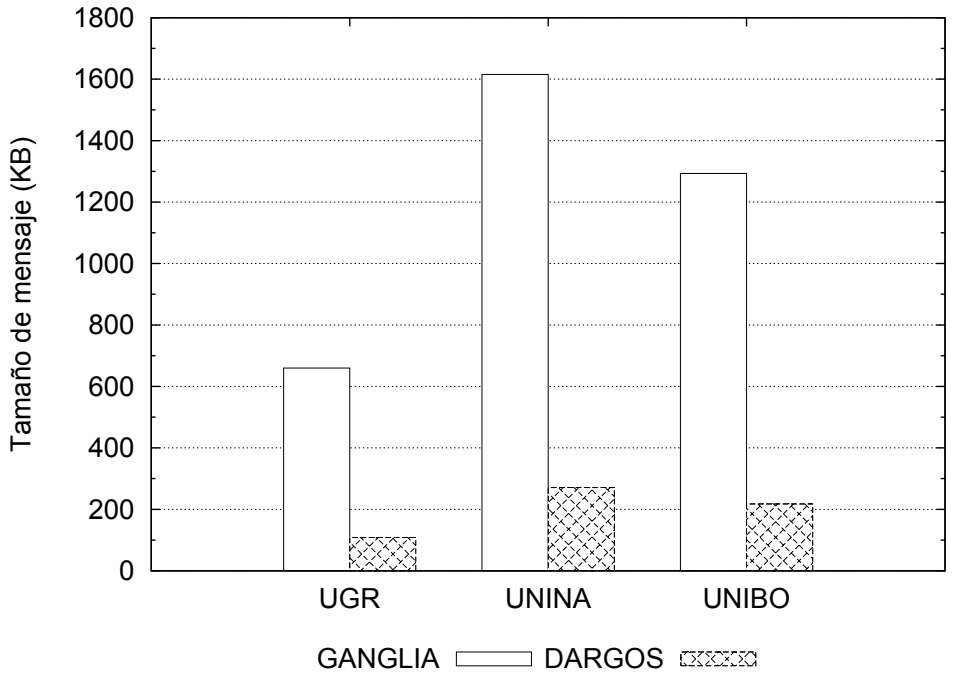


Figura 4.19: Comparación DARGOS vs. *Ganglia* en WAN: tamaño de mensajes

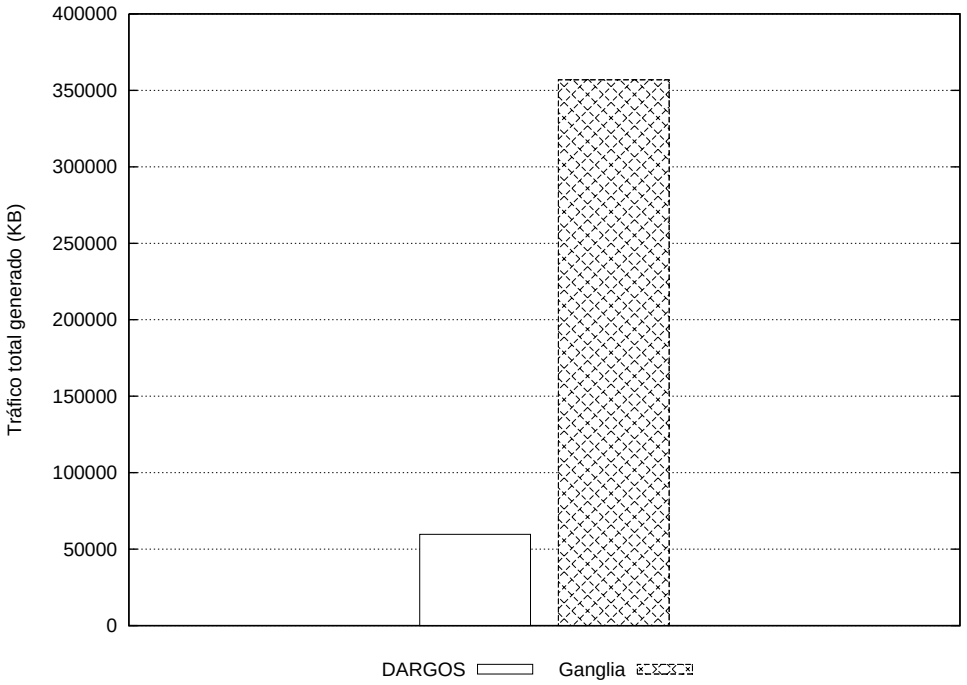


Figura 4.20: Comparación DARGOS vs. *Ganglia* en WAN: tráfico total generado

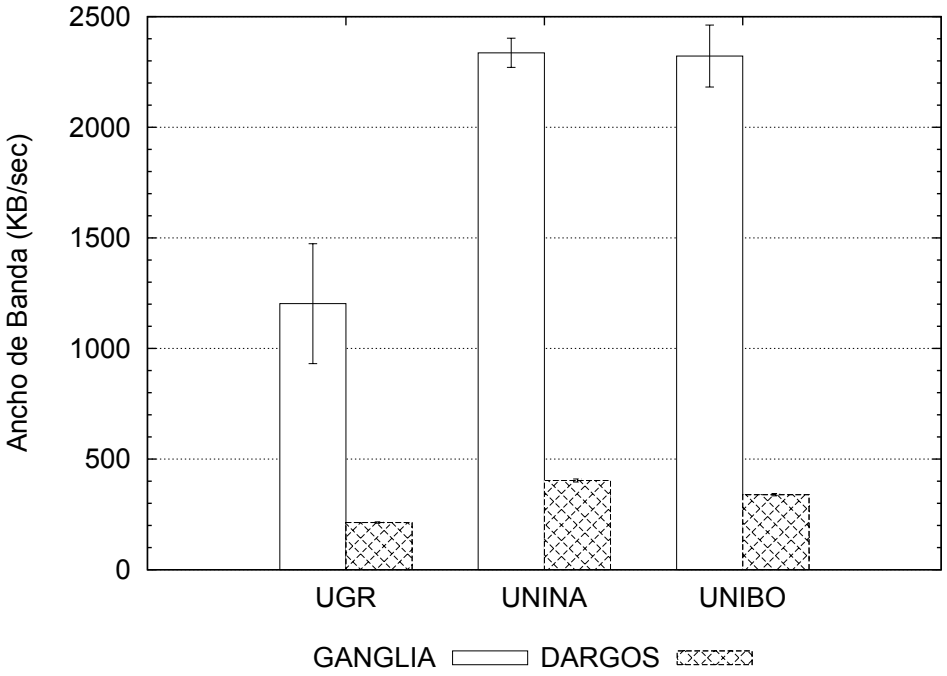


Figura 4.21: Comparación DARGOS vs. *Ganglia* en WAN: ancho de banda

5 Conclusiones

En este último capítulo se resumen las principales conclusiones obtenidas tras la realización de esta Tesis. Además se detallan algunas posibles líneas de trabajo futuro.

5.1. Conclusiones

El procesamiento distribuido y la provisión de servicios para lujos multimedia en tiempo real es un campo que está ganando cada vez más interés debido a la aparición de dispositivos más potentes. Típicamente, este tipo de aplicaciones han sido implementadas por medio de soluciones centralizadas, en la que uno o varios servidores procesan y distribuyen los flujos multimedia bajo demanda. Sin embargo, este paradigma presenta una serie de inconvenientes, tales como la escalabilidad, la aparición de cuellos de botella y puntos centralizados de fallo.

Gran parte de los inconvenientes de esta aproximación centralizada es debida al acoplamiento existente entre los servidores que proporcionan los servicios y los clientes consumidores de los correspondientes flujos. Para resolver estos problemas, en los últimos años han hecho aparición paradigmas tales como el paradigma de publicación/suscripción. Este elimina el acoplamiento existente entre productores y consumidores de información.

Dentro de la aproximación publicación/suscripción existen distintas variantes. Una de ellas es la aproximación publicación/suscripción centrada en datos. En esta aproximación, los intercambios de información entre publicadores y suscriptores vienen condicionados por el contenido de los datos publicados, permitiendo así de una manera sencilla operaciones como el filtrado y el almacenamiento temporal de la información en memoria cache.

El paradigma de publicación/suscripción centrada en datos ha demostrado su eficiencia en escenarios muy exigentes como mercados financieros o entornos militares. En esta Tesis se ha realizado un estudio de los beneficios que este paradigma proporciona a los sistemas distribuidos de procesado y distribución de contenido multimedia.

En el Capítulo 3 se ha estudiado cómo se puede utilizar el paradigma de publicación/suscripción centrada en datos en sistemas de procesamiento multimedia distribuidos. Para ello se ha propuesto EMDS, una solución que utiliza dicho paradigma.

EMDS permite el despliegue de servicios multimedia desacoplados tanto de los productores de información como de los consumidores de la misma. Dichos servicios se pueden utilizar para analizar flujos multimedia, procesarlos y/o transformarlos. EMDS se ha diseñado utilizando un enfoque basado en componentes. Estos pueden interaccionar cooperando entre si para así construir distintos servicios que pueden ser empleados en diferentes escenarios. Además de los componentes que componen EMDS, se ha realizado un estudio de cómo implementar diversos tipos de servicios utilizando dichos componentes. Gracias al enfoque adoptado en EMDS, los servicios implementados son capaces de descubrir los flujos publicados y convertirse a su vez en nuevos orígenes de datos multimedia basados en los flujos recibidos que pueden ser nuevamente procesados y consumidos.

Como características más relevantes de EMDS destaca su capacidad de integración con sistemas ya existentes y su extensibilidad, ya que permite añadir nuevos servicios automáticamente y de una manera transparente al usuario. Esto permite su uso en escenarios de diversa índole que abarcan desde sistemas de vídeo-vigilancia hasta sistemas de colaboración remota [114].

Un prototipo de EMDS se ha implementado para validar y evaluar su diseño. Utilizando dicho prototipo, tras la experimentación realizada se han presentado resultados que ponen de manifiesto la facilidad para construir servicios de una manera sencilla utilizando componentes EMDS. Además bajo las condiciones descritas EMDS reduce el consumo de ancho de banda y el retardo introducido frente a otras aproximaciones de referencia. Para ello se ha comparado la eficiencia de nuestra propuesta frente a un sistema centralizado basado en RTP y otro basado en AMQP. Los resultados muestran las ventajas tanto cualitativas como cuantitativas que reporta el uso de una aproximación publicación/suscripción centrada en datos.

Otro aspecto que se ha estudiado en esta Tesis es la monitorización de recursos en entornos distribuidos. En el Capítulo 4 se ha presentado DARGOS, un sistema de monitorización en tiempo real de recursos. El diseño de DARGOS permite monitorizar tanto recursos software como hardware de una manera escalable y eficiente. Gracias a DARGOS las aplicaciones de monitorización pueden descubrir de manera automática los recursos disponibles y personalizar los requisitos de recepción de

actualizaciones de uso por parte de la aplicación.

DARGOS ha sido diseñado para la monitorización de recursos en entornos altamente dinámicos y con requisitos importantes de tiempo real, tales como la monitorización de sistemas multimedia distribuidos. Sin embargo, su flexibilidad y generalidad le permite ser empleado en múltiples escenarios, tales como la gestión de recursos en entornos *Cloud Computing*.

Gracias a DARGOS es posible identificar problemas tales como sobrecarga de recursos, para así permitir a los administradores iniciar acciones para solventar dichos problemas. Así por ejemplo, DARGOS se puede utilizar para identificar sobrecargas en servicios desplegados en EMDS y en consecuencia iniciar acciones que permitan una mejor gestión de los recursos disponibles mediante balanceo de carga o migración de un servicio EMDS sobrecargado a otro nodo mejor.

Se ha implementado un prototipo de DARGOS para su validación y evaluación. Concretamente, se muestran resultados experimentales que demuestran cómo la aproximación publicación/suscripción centrada en datos empleada por DARGOS reduce el consumo de ancho de banda frente a otros servicios de monitorización disponibles en la bibliografía. Como consecuencia de esta reducción de recursos de red, se demuestra una mejora en la escalabilidad del sistema propuesto.

Para finalizar esta sección, se realizará un análisis *a posteriori* de la consecución de los objetivos de esta Tesis enumerados en el capítulo 1

- Provisión de servicios, Gestión y distribución de contenido multimedia.

  Objetivo. Diseñar un sistema para la provisión de servicios, procesamiento y distribución de contenido multimedia en tiempo real basado en la aproximación publicación/suscripción centrada en datos.

  - Se ha diseñado EMDS un sistema que permite el procesado y la provisión de servicios distribuidos de flujos multimedia. Los resultados de este objetivo han sido presentados en diversos foros, destacando esta la publicación en *Real-time and Embedded Systems Workshop 2010* [145].

  Objetivo. Facilitar el desarrollo de sistemas multimedia distribuidos.

  - La arquitectura basada en componentes propuesta por EMDS hace posible desarrollar y desplegar servicios de una manera sencilla, gracias a la aproximación centrada en datos empleada.

  Objetivo. Implementar y evaluar las prestaciones del sistema diseñado.

  - Se ha implementado un prototipo de EMDS, que además ha sido evaluado en términos de retardo y ancho de banda frente a otras alternativas empleadas para la distribución de contenido multimedia.

- Monitorización de recursos distribuidos.

  Objetivo. Diseñar un sistema de monitorización de recursos en tiempo real basado en la aproximación publicación/suscripción centrada en datos.

  - Se ha diseñado DARGOS un sistema que permite la monitorización de recursos en tiempo real en entornos distribuidos altamente dinámicos.

  Objetivo. Facilitar la gestión de sistemas distribuidos.

  - DARGOS permite el descubrimiento automático de los recursos disponibles en el sistema. La aproximación publicación/suscripción empleada además permite que cada suscriptor pueda decidir qué información recibir y la frecuencia con la que la quiere recibir.

  Objetivo. Implementar y evaluar el sistema diseñado comparándolo con otros sistemas de referencia.

  - Se ha desarrollado un prototipo de DARGOS que ha sido comparado con otros sistemas de amplia difusión disponibles en la bibliografía. Se ha observado que DARGOS reduce el consumo de ancho de banda además de facilitar el despliegue. Los resultados de esta implementación y evaluación han sido publicados en una revista indexada en JCR [146].

5.2. Trabajo futuro

Como trabajo futuro a esta Tesis, una de las principales líneas de investigación es la integración de EMDS y DARGOS en entornos Cloud Computing. Esta integración permitirá la construcción de servicios multimedia personalizados en la nube de una manera casi automática y reduciendo así los costes asociados con el despliegue y mantenimiento de una infraestructura.

En el caso de EMDS consideramos que una línea de investigación abierta importante es la implementación de mecanismos de migración de servicios *EMDS* a distintos nodos de manera automatizada, permitiendo así la optimización de recursos mediante técnicas de consolidación o balanceo de carga [34, 36, 188, 192].

En el caso de DARGOS además consideramos que existen dos líneas abiertas de investigación. La primera de ellas consiste en la integración con estándares de monitorización en *Cloud* [33] además de la integración con otros sistemas de monitorización existentes ampliamente difundidos como *Ganglia* [116], haciendo de DARGOS una herramienta que integre la información capturada en despliegues ya existentes. La segunda línea considerada es el estudio de mecanismos de gestión que permitan convertir a DARGOS en una herramienta de gestión completa para la gestión de sistemas de manera automatizada y desatendida.

5 Conclusions

In this Chapter, we discuss the main conclusions obtained from our work. In addition, we also consider some future work directions.

5.1. Conclusions

Distributed processing of multimedia streams in real time is a field that is gaining momentum due to the appearance of more and more powerful devices. Typically, such kind of applications have been developed using centralized solutions. That is, one or multiple well known servers process and deliver multimedia streams upon request. However, this paradigm has inherent drawbacks, such as scalability, bottleneck and centralized point of failure.

Most of the drawbacks of centralized approaches are due to the strong coupling between the servers –which are providing the service– and the clients –which are consuming the streams–. To cope with these drawbacks, some paradigms such as the publish/subscribe approach have been recently proposed. They remove the existing coupling between the data producers and consumers.

Depending on how the communication between publishers and subscribers is done, multiple varieties of this paradigm can be identified. One of them is the data centric publish/susbcribe approach. In this approach, data exchanges between publishers and subscribers are conditioned by the content of the data itself, thus making possible to implement easily operations such as data filtering and caching.

The data-centric publish/subscribe paradigm has demonstrated its efficiency in very demanding scenarios such as stock market or military environments. In this Thesis we make a study on the benefits reported by using it in the building of mul-

timedia distribution and processing systems.

In Chapter 3 we have studied how the data centric publish/subscribe approach can be used in distributed multimedia processing systems. For this purpose, we have proposed EMDS, a solution based in this paradigm that allows to build multimedia systems for data processing and distribution.

EMDS allows the deployment of multimedia services that are completely decoupled both from data sources and destinations. Such services can be used to analyze, process and transform multimedia streams. EMDS has been designed as a set of components that cooperate to build different multimedia services. Apart from the EMDS components, we have carried out a study on how to implement different kind of services based on such components. Thanks to the approach used in our proposal, the implemented services are able to discover available streams and to process them to generate new streams that can be processed and consumed by other services/subscribers.

A relevant EMDS characteristics is it's integration capacity with legacy systems and its extensibility, as it can add new services automatically and in a non disruptive manner. This feature allows EMDS to be used in diverse scenarios ranging from video surveillance systems to remote collaboration environments [147].

We have developed a prototype of EMDS to assess our proposal. Using this prototype we presented some experimental evaluations. The results show that its component based approach –apart from facilitating the provision of multimedia services– reduces the bandwidth and delay. We have compared EMDS with other approaches such as a RTP centralized scheme and a AMQP multimedia service as well. The results show that the advantages on using a data centric approach are not only qualitative (*i.e.* ease of deployment, extensibility, etc.) but also quantitative.

Another aspect studied in this Thesis is the real time resource monitoring in distributed environments. In Chapter 4 we have presented DARGOS, a system for monitoring resources in real time using a data centric publish/subscribe approach. DARGOS design allows to monitor software and hardware resources in a scalable manner and efficient in terms of bandwidth usage. Thanks to DARGOS, monitoring applications are able to discover available resources and customize their update settings.

DARGOS has been designed for resource monitoring in highly dynamic environments with important requirements in real time, such as the monitoring of distributed multimedia environment. However, its flexibility and generality enable it to be used in multiple scenarios, such as resource management in *Cloud Computing* environments [37, 146].

DARGOS eases the identification of problems in distributed environment such

as resource overload, Thus, it allows system administrator starting actions to solve such problems. For instance, DARGOS can be used to identify overload in EMDS services to trigger actions that allows better resource management such as load balancing or migrating the EMDS services to a more powerful hosts. Thanks to the data-centric approach, such actions occur in a transparent manner to the final user as the services are not associated with a physical address, as usually happens in server/client approaches.

We implemented a DARGOS prototype in order to assess the data centric approach in resource monitoring. The conducted experimental evaluation demonstrates how DARGOS reduces the bandwidth usage in comparison with other monitoring reference systems. As a consequence of this network usage minimization, DARGOS improves the overall scalability of the system.

To finalize this Section, we revise the level of goal achieving as they were formulated in Chapter 1:

- Distribution and management of multimedia content

  Goal: Design a system to process and distribute multimedia content in real time based in the data-centric publish/subscribe approach.

  - We have proposed EMDS, a system for providing distributed multimedia services. This contribution has been presented in multiple forums, for instance in the *Real-time and Embedded Systems Workshop 2010* [145].

  Goal: Easing the development of distributed multimedia systems.

  - The proposed EMDS component-based approach makes possible to develop and deploy multimedia services in an easy way. This is also possible to the data centric approach adopted, given that it removes the decoupling between data publishers and subscribers.

  Goal: To implement and evaluate the capabilities of the designed system.

  - We have implemented an EMDS prototype, that has been evaluated in terms of delay and bandwidth.

- Distributed resource monitoring

  Goal: To Design a real time resource monitoring system based in the data-centric publish/suscribe paradigm.

  - We have designed DARGOS a system that allows resources monitoring in real time for highly dynamic environments.

  Goal: To ease the management and deployment of distributed systems.

  - DARGOS allows the automatic discovery of the resources available in the system. The publish/subscribe approach also allows choosing which information to receive and the update rate.

Goal: To implement and evaluate the proposed system in comparison with other reference systems.

- We have developed a DARGOS prototype that has been compared with other reference monitoring systems. The experimental results show that DARGOS eases the deployment, reduces the bandwidth and, increases the scalability. This research has been published in an indexed (JCR) journal [146].

5.2. Future work

As future work for this Thesis, one of the main research directions is the integration of EMDS and DARGOS in Cloud Computing environments. Such integration will make possible to build customized multimedia services in an automatic fashion while reducing the cost associated to the deployment and maintenance of the data center.

In the case of EMDS we consider that another open research direction is the implementation of non attended EMDS service migration mechanisms. It will allow for instance resource usages optimization with load balancing and consolidation techniques [34, 36, 188, 192].

In the case of DARGOS we consider that there are two main open research directions. The first one refers to the integration of DARGOS with other Cloud monitoring standards [33]. In this respect, another challenge to be coped would be DARGOS integration with other popular monitoring systems such as *Ganglia*, thus making it an integrating tool that captures monitoring data in already deployed systems. The second research direction is the study of management mechanisms that allow DARGOS to become a complete management tool capable of manage distributed systems in a non attended and automatized way.

Bibliografía

[1] ACM. Cloud Computing: An Overview. *Distributed Computing*, 7(5):1–5, 2009.

[2] Paulina Adamska, Adam Wierzbicki, and Tomasz Kaszuba. A Generic and Overlay-Agnostic Publish-Subscribe Protocol. *2009 First International Conference on Advances in P2P Systems*, pages 98–103, October 2009. doi: 10.1109/AP2PS.2009.41. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5359068.

[3] M Ahmed and AM Mansor. CPU dimensioning on performance of Asterisk VoIP PBX. *Proceedings of the 11th communications and ...*, 2008. URL http://dl.acm.org/citation.cfm?id=1400737.

[4] B. Almadani. RTPS Middleware for Real-Time Distributed Industrial Vision Systems. In *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*, pages 361–364. Ieee, 2005. ISBN 0-7695-2346-3. doi: 10.1109/RTCSA.2005.85. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1541108.

[5] Michael Altenhofen, Jürgen Dittrich, Rainer Hammerschmidt, Thomas Käppner, Carsten Kruschel, Ansgar Kückes, and Thomas Steinig. The BERKOM multimedia collaboration service. *Proceedings of the first ACM international conference on Multimedia - MULTIMEDIA '93*, pages 457–463, 1993. doi: 10.1145/166266.168460. URL http://portal.acm.org/citation.cfm?doid=166266.168460.

[6] MB Amin, WA Khan, AA Awan, and S Lee. Intercloud message exchange middleware. *Proceedings of the 6th ...*, 2012. URL http://dl.acm.org/citation.cfm?id=2184845.

[7] Emil Andriescu, Roberto Speicys Cardoso, and V Issarny. AmbiStream: a middleware for multimedia streaming on heterogeneous mobile devices. *Middleware 2011*, 7049, 2011. URL http://www.springerlink.com/index/0W1U13M022935387.pdf.

[8] Apache Foundation. ActiveMQ, 2013.

[9] DI Applebaum. The Galatea Network Video Device Control System. Technical report, Massachusetts Institute of Technology, 1991. URL http://mf.media.mit.edu/pubs/techreport/ProjGalatea.pdf.

[10] G Banavar, T Chandra, R Strom, and D Sturman. A case for message oriented middleware. *Distributed Computing*, 1999. URL http://link.springer.com/chapter/10.1007/3-540-48169-9_1.

[11] J Bankoski, J Koleszar, L Quillio, J Salonen, P Wilkins, and Y Xu. VP8 Data Format and Decoding Guide. RFC 6386 (Informational), November 2011. URL http://www.ietf.org/rfc/rfc6386.txt.

[12] Kai Beckmann and Marcus Thoss. A wireless sensor network protocol for the OMG Data Distribution Service. *... WISES), 2012 Proceedings of the Tenth ...*, pages 45–50, 2012. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6273603.

[13] T Benson, A Akella, and DA Maltz. Network traffic characteristics of data centers in the wild. *Proceedings of the 10th ACM SIGCOMM ...*, 2010. URL http://dl.acm.org/citation.cfm?id=1879175.

[14] E Bonaccorsi and N Neufeld. Monitoring the LHCb experiment computing infrastructure with NAGIOS. *Proceedings of ICALEPCS 2009*, 2009. URL http://cdsweb.cern.ch/record/1215281.

[15] Pruet Boonma and Junichi Suzuki. Toward interoperable publish/subscribe communication between wireless sensor networks and access networks. In *... Conference, 2009. CCNC 2009. 6th IEEE*, pages 1–6, 2009. ISBN 9781424423095. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4784935.

[16] CM Bray, Tim and Paoli, Jean and Sperberg-McQueen. Extensible markup language (XML) 1.0., 1998.

[17] A Broder and M Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 2004. URL http://www.tandfonline.com/doi/full/10.1080/15427951.2004.10129096.

[18] PhD Bruce Tolley. 10Gb Ethernet: The Foundation for Low-Latency, Real-Time Financial Services Applications and Other, Latency-Sensitive Applications. Technical report, Solarflare Inc., 2013. URL http://www.solarflare.com/Content/UserFiles/Documents/Arista_Solarflare_Low_Latency_10GbE.pdf.

[19] Hasan Bulut, Geoffrey Fox, Shrideep Pallickara, Ahmet Uyar, and Wenjun Wu. Integration of NaradaBrokering and Audio/Video conferencing as a web service. *Proceedings of the IASTED International Conference on Communications, Internet, and Information Technology*, pages 18–20, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.3979&rep=rep1&type=pdfhttp://surface.syr.edu/eecs/48/.

[20] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, June 2009. ISSN 0167739X. doi: 10.1016/j.future.2008.12.001. URL http://linkinghub.elsevier.com/retrieve/pii/S0167739X08001957.

[21] Lorena Calavia, Carlos Baladrón, Javier M Aguiar, Belén Carro, and Antonio Sánchez-Esguevillas. A semantic autonomous video surveillance system for dense camera networks in Smart Cities. *Sensors (Basel, Switzerland)*, 12(8):10407–29, January 2012. ISSN 1424-8220. doi: 10.3390/s120810407. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3472835&tool=pmcentrez&rendertype=abstract.

[22] Jiannong Cao, Xuefeng Liu, Steven Lai, Yang Zou, Jun Zhang, Yang Liu, and Chisheng Zhang. A ubiquitous wireless video surveillance system based on pub/sub. *Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11*, page 611, 2011. doi: 10.1145/2030112.2030239. URL http://dl.acm.org/citation.cfm?doid=2030112.2030239.

[23] B Carpenter. Architectural Principles of the Internet. RFC 1958 (Informational), June 1996. URL http://www.ietf.org/rfc/rfc1958.txt.

[24] J Case, K McCloghrie, M Rose, and S Waldbusser. Introduction to version 2 of the Internet-standard Network Management Framework. RFC 1441 (Historic), April 1993. URL http://www.ietf.org/rfc/rfc1441.txt.

[25] J D Case, M Fedor, M L Schoffstall, and J Davin. Simple Network Management Protocol (SNMP). RFC 1157 (Historic), May 1990. URL http://www.ietf.org/rfc/rfc1157.txt.

[26] C Chaudet, I Demeure, and S Ktari. Publish/subscribe for wireless sensor networks. *Proceedings of the 7th ...*, 2011. URL http://dl.acm.org/citation.cfm?id=2089020.

[27] J Choi, J Han, and E Cho. A survey on content-oriented networking for efficient content delivery. *Communications ...*, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5723809.

[28] Wu-Chun Chung and Ruay-Shiung Chang. A new mechanism for resource monitoring in Grid computing. *Future Generation Computer Systems*, 25(1):1–7, January 2009. ISSN 0167739X. doi: 10.1016/j.future.2008.04.008. URL http://dx.doi.org/10.1016/j.future.2008.04.008http://linkinghub.elsevier.com/retrieve/pii/S0167739X0800054X.

[29] S Clayman, A Galis, and L Mamatas. Monitoring virtual networks with Lattice. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 239–246. Dept of Electron. Eng., Univ. Coll. London, London, UK, IEEE, 2010. ISBN 978-1-4244-6037-3. doi: 10.1109/NOMSW.2010.5486569. URL http://dx.doi.org/10.1109/NOMSW.2010.5486569.

[30] S Clayman, R Clegg, L Mamatas, G Pavlou, and A Galis. Monitoring, aggregation and filtering for efficient management of virtual networks. In *Network and Service Management (CNSM), 2011 7th International Conference on*, pages 1–7. Dept. of Electron. Eng., Univ. Coll. London, London, UK, IEEE, 2011. ISBN 978-1-4577-1588-4. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6103978.

[31] Stuart Clayman, Alex Galis, Clovis Chapman, Giovanni Toffetti, Luis Rodero-Merino, L.M. Luis M Vaquero, Kenneth Nagin, and Benny Rochwerger. Monitoring service clouds in the future internet. *Towards the Future Internet-Emerging Trends from European Research*, 0:115, 2010. ISSN 978-1-60750-538-9. doi: 10.3233/978-1-60750-539-6-115. URL http://www.future-internet.eu/fileadmin/documents/valencia_documents/plenary/Monitoring_service_clouds_in_the_Future_Internet.pdf.

[32] Stuart Clayman, Alex Galis, Clovis Chapman, Giovanni Toffetti, Luis Rodero-Merino, L.M. Luis M Vaquero, Kenneth Nagin, and Benny Rochwerger. Lattice Monitoring Framework, 2012. URL http://clayfour.ee.ucl.ac.uk/lattice/.

[33] Cloud Management Working Group. Cloud Infrastructure Management Interface (CIMI) Model and REST Interface over HTTP Specification. Technical report, Distributed Management Task Force, 2012.

[34] A Corradi, M Fanelli, and L Foschini. VM consolidation: A real case based on openstack cloud. *Future Generation Computer Systems*, 2012. URL http://www.sciencedirect.com/science/article/pii/S0167739X12001082.

[35] Antonio Corradi and Luca Foschini. A DDS-compliant P2P infrastructure for reliable and QoS-enabled data dissemination. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009. URL http://www.computer.org/portal/web/csdl/doi/10.1109/IPDPS.2009.5160957http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5160957.

[36] Antonio Corradi, Franco Zambonelli, and Letizia Leonardi. Diffusive Load-Balancing Policies for Dynamic Applications. *IEEE Concurrency*, 1999.

[37] Antonio Corradi, Luca Foschini, Javier Povedano-Molina, and Juan M. Lopez-Soler. DDS-enabled Cloud management support for fast task offloading. *2012 IEEE Symposium on Computers and Communications (ISCC)*, pages 000067–000074, July 2012. doi: 10.1109/ISCC.2012.6249270. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6249270.

[38] Angelo Corsaro and Leonardo Querzoni. Quality of service in publish/subscribe middleware. *Global Data ...*, pages 1–19, 2006. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Quality+of+Service+in+Publish/Subscribe+Middleware#1http://www.omgwiki.org/dds/sites/default/files/Quality_of_Service_in_Publish-Subscribe.pdf.

[39] D Crockford. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 (Informational), July 2006. URL http://www.ietf.org/rfc/rfc4627.txt.

[40] T Crowley, P Milazzo, and E Baker. MMConf: An infrastructure for building shared multimedia applications. *Proceedings of the ...*, 1990. URL http://dl.acm.org/citation.cfm?id=99365.

[41] F Curbera, M Duftler, and R Khalaf. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *Internet Computing ...*, 2002. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=991449.

[42] S.A. De Chaves, R.B. Uriarte, and C.B. Westphall. Toward an architecture for monitoring private clouds. *Communications Magazine, IEEE*, 49(12): 130–137, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6094017.

[43] JD Decotignie and P Dallemagne. Expressing Audio/Video communications using the publish-subscribe model. In *The 2004 ECRTS Workshop on Real-time*

for Multimedia (RTMM), pages 3–74, 2004. URL http://alexandria.tue.nl/extra1/wskrap/publichtml/200436.pdfhttp://alexandria.tue.nl/extra1/wskrap/public_html/200436.pdf#page=21.

[44] L Delgrossi and J Sandvoss. The BERKOM-II multimedia transport system, 1993. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+BERKOM+Multimedia+Transport+System#1.

[45] Andrea Detti, Pierpaolo Loreti, N. Blefari-Melazzi, and Francesco Fedi. Streaming H. 264 scalable video over data distribution service in a wireless environment. In *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*, pages 1–3. IEEE, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5534937.

[46] T Dierks and C Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. URL http://www.ietf.org/rfc/rfc2246.txt.

[47] T Dierks and E Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. URL http://www.ietf.org/rfc/rfc5246.txt.

[48] D Eastlake 3rd. HMAC SHA (Hashed Message Authentication Code, Secure Hash Algorithm) TSIG Algorithm Identifiers. RFC 4635 (Proposed Standard), August 2006. URL http://www.ietf.org/rfc/rfc4635.txt.

[49] D Eastlake 3rd and T Hansen. US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). RFC 6234 (Informational), May 2011. URL http://www.ietf.org/rfc/rfc6234.txt.

[50] D Eastlake 3rd and P Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174 (Informational), September 2001. URL http://www.ietf.org/rfc/rfc3174.txt.

[51] M Eisler. XDR: External Data Representation Standard. RFC 4506 (Standard), May 2006. URL http://www.ietf.org/rfc/rfc4506.txt.

[52] James Elder. Geoide ProjecT PIV - 17, 2012. URL http://gold.geoide.ulaval.ca/upload/projets/PIV17_eng.pdf.

[53] C Eryigit and S Uyar. Integrating agents into data-centric naval combat management systems. *...and Information Sciences, 2008. ISCIS'08. ...*, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4717890.

[54] I Etxeberria-Agiriano. Providing soft real-time capabilities to business applications. *Information Systems and ...*, 2012. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6263138.

[55] P Eugster. Type-based publish/subscribe: Concepts and experiences. *ACM Transactions on Programming Languages and ...*, 2007. URL http://dl.acm.org/citation.cfm?id=1180481.

[56] P.T. Eugster, P.A. Felber, R. Guerraoui, and A.M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003. URL http://portal.acm.org/citation.cfm?id=857078.

[57] D Ferrari and S Zhou. *An empirical investigation of load indices for load balancing applications*. Defense Technical Information Center., 1987. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/1987/CSD-87-353.pdf.

[58] B Fitzpatrick, B Slatkin, and M Atkins. PubSubHubbub Core 0.3–working draft. *.... com/svn/trunk/pubsubhubbub-core-0.3. html*, 2010. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:PubSubHubbub+Core+0.3?working+draft#0.

[59] T Fitzpatrick, J Gallop, and G Blair. Design and application of TOAST: an adaptive distributed multimedia middleware platform. *...Distributed Multimedia ...*, 2001. URL http://link.springer.com/chapter/10.1007/3-540-44763-6_13.

[60] G.C. Fox, Wenjun Wu, Ahmet Uyar, and Hasan Bulut. Design and implementation of audio/video collaboration system based on publish/subscribe event middleware. In *Proceedings of CTS04 San Diego January*. Citeseer, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.1394&rep=rep1&type=pdfhttp://surface.syr.edu/eecs/132/.

[61] G.C. Fox, M. Pierce, and H. Bulut. eSports: Collaborative and Synchronous Video Annotation System in Grid Computing Environment. *Seventh IEEE International Symposium on Multimedia (ISM'05)*, pages 95–103, 2005. doi: 10.1109/ISM.2005.55. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1565818.

[62] Geoffrey Fox and Shrideep Pallickara. The Narada event brokering system: Overview and extensions. *... of the International Conference on ...*, 2002. URL http://www.naradabrokering.org/papers/NaradaBrokeringSystem.pdf.

[63] E. Galstad. *Nagios: Nrpe documentation.* Sourceforge.net, 2007.

[64] Ganglia Project. Ganglia Monitoring System, 2013. URL http://ganglia.info.

[65] M. Garcia-Valls, P. Basanta-Val, and I. Estevez-Ayres. Adaptive real-time video transmission over DDS. In *Industrial Informatics (INDIN), 2010 8th*

IEEE International Conference on, pages 130–135. IEEE, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5549450.

[66] AR Girard, AS Howell, and JK Hedrick. Border patrol and surveillance missions using multiple unmanned air vehicles. *Decision and Control, 2004 ...*, 2004. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1428713.

[67] Google Inc. Google Container Data Center Tour, 2009. URL http://www.youtube.com/watch?v=zRwPSFpLX8I.

[68] Google Inc. Google Hangouts, 2013. URL http://www.google.com/intl/es_ALL/+/learnmore/hangouts/.

[69] R Gordon. *Essential JNI: Java Native Interface*. Prentice Hall PTR, 1998. URL http://dl.acm.org/citation.cfm?id=SERIES9932.275573.

[70] Gstreamer Project. Gstreamer Multimedia Framework, 2013. URL http://gstreamer.freedesktop.org/.

[71] X Gu and K Nahrstedt. Distributed multimedia service composition with statistical QoS assurances. *Multimedia, IEEE Transactions on*, 2006. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1580533.

[72] Abhishek Gupta, O.D. Sahin, Divyakant Agrawal, and A.E. Abbadi. Meghdoot: content-based publish/subscribe over P2P networks. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 254–273. Springer-Verlag New York, Inc., 2004. URL http://dl.acm.org/citation.cfm?id=1045677.

[73] Mark Hapner and Rich Burridge. Java Message Service. Technical report, Sun Microsystems Inc., 2002. URL https://download.oracle.com/otn-pub/jcp/jms-2_0-edr-spec/jms-2_0-edr-spec.pdf.

[74] D Harrington, R Presuhn, and B Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411 (Standard), December 2002. URL http://www.ietf.org/rfc/rfc3411.txt.

[75] P. Hasselmeyer and N D'Heureuse. Towards holistic multi-tenant monitoring for virtual data centers. *Network Operations and ...*, pages 350–356, 2010. doi: 10.1109/NOMSW.2010.5486528. URL http://dx.doi.org/10.1109/NOMSW.2010.5486528http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5486528http://surface.syr.edu/eecs/48/.

[76] R Hinden. Host Monitoring Protocol. RFC 869 (Historic), December 1983. URL http://www.ietf.org/rfc/rfc869.txt.

[77] He Huang and Liqiang Wang. P&P: A Combined Push-Pull Model for Resource Monitoring in Cloud Computing Environment. *2010 IEEE 3rd International Conference on Cloud Computing*, pages 260–267, July 2010. doi: 10.1109/CLOUD.2010.85. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5557986http://dx.doi.org/10.1109/CLOUD.2010.85.

[78] Yuanqiang Huang, Zhongzhi Luan, Depei Qian, Jiali Du, Xiang Ni, and Lan Gao. NovaPS: A robust Publish/Subscribe overlay for multimedia communication. *Proceedings of the 5th International ICST Conference on Communications and Networking in China*, 2010. doi: 10.4108/chinacom.2010.99. URL http://eudl.eu/doi/10.4108/chinacom.2010.99.

[79] Zixia Huang, Chao Mei, L.E. Li, and Thomas Woo. CloudStream: delivering high-quality streaming videos through a cloud-based SVC proxy. In *INFOCOM, 2011 Proceedings IEEE*, pages 201–205. IEEE, 2011. ISBN 9781424499212. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5935009.

[80] Kilroy Hughes. The Future of Cloud-Based Entertainment. *Proceedings of the IEEE*, 100(Special Centennial Issue):1391–1394, May 2012. ISSN 0018-9219. doi: 10.1109/JPROC.2012.2189790. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6179962http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6179962.

[81] U Hunkeler. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. Technical report, IBM/Eurotech, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4554519.

[82] E Imamagic and D Dobrenic. Grid infrastructure monitoring system based on Nagios. *...of the 2007 workshop on Grid monitoring*, 2007. URL http://dl.acm.org/citation.cfm?id=1272680.1272685.

[83] International Business Machines Corporation and EUROTECH. MQTT V3.1 Protocol Specification. Technical report, IBM/Eurotech, 2010. URL http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/MQTT_V3.1_Protocol_Specific.pdf.

[84] ITU-T. G. 729: Coding of speech at 8 kbit/s using conjugate structure algebraic-code-excited linear-prediction (CS-ACELP). *Recommendation G*, 1996.

[85] ITU-T. H. 262: Information Technology–Generic Coding of Moving Pictures and Associated Audio Information: Video. *Recommendation H*, 1996. URL http://www.itu.int/rec/T-REC-H.262.

[86] ITU-T. H.323: Packet-based multimedia communications systems. *Recommendation H*, 1996. URL http://www.itu.int/rec/T-REC-H.323/en/.

[87] ITU-T. H. 263: Video coding for low-bit-rate communication. *Recommendation H*, 1997. URL http://www.itu.int/rec/T-REC-H.263/.

[88] ITU-T. T.140: Protocol for multimedia application text conversation. *Recommendation T*, 1998.

[89] ITU-T. H.264 (11/2007). *Group*, 2007.

[90] Van Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12, New York, New York, USA, 2009. ACM. ISBN 9781605586366. doi: 10.1145/1658939.1658941. URL http://portal.acm.org/citation.cfm?id=1658941.

[91] JBOSS Community. HornetQ, 2013. URL http://www.jboss.org/hornetq/.

[92] Z Jerzak. *XSiena: The Content-Based Publish/Subscribe System*. Phd. dissertation, University of Dresden, 2009. URL http://jerzak.eu/publ/doc/jerzak2009xsiena.pdf.

[93] A Jones and A Hopper. Handling audio and video streams in a distributed environment. *ACM SIGOPS Operating Systems Review*, 1994. URL http://dl.acm.org/citation.cfm?id=168638.

[94] Jeffrey Joyce, Greg Lomow, Konrad Slind, and Brian Unger. Monitoring distributed systems. *Transactions on Computer Systems (TOCS)*, 5(2):121–150, March 1987. ISSN 07342071. doi: 10.1145/13677.22723. URL http://portal.acm.org/citation.cfm?doid=13677.22723http://dl.acm.org/citation.cfm?id=22723.

[95] JSON.org. JSON, 2013. URL http://json.org/.

[96] HN Karaca, HM Yüksel, and E Tüzün. A Data-Centric Framework for Network Enabled C4ISR Software Systems. *... and Information Sciences*, 2010. URL http://link.springer.com/chapter/10.1007/978-90-481-9794-1_34.

[97] K Katsaros and N Fotiou. Supporting mobile streaming services in future publish/subscribe networks. *..., 2009. WTS 2009*, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5068987.

[98] S Kent and K Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005. URL http://www.ietf.org/rfc/rfc4301.txt.

[99] R Khare and RN Taylor. Extending the representational state transfer (REST) architectural style for decentralized systems. *Software Engineering, 2004. ICSE 2004. ...*, 2004. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1317465.

[100] GK Kloss. Automatic C Library Wrapping Ctypes from the Trenches. *The Python Papers*, 2008. URL http://ojs.pythonpapers.org/index.php/tpp/article/viewArticle/71.

[101] Christopher Kohlhoff and Robert Steele. Evaluating SOAP for high performance business applications: Real-time trading systems. *Proceedings of WWW*, 2003. URL http://pdf.aminer.org/000/674/302/evaluating_soap_for_high_performance_business_applications_real_time_trading.pdf.

[102] Benjamin König, Jose M. Alcaraz Calero, and Johannes Kirshnick. Elastic Monitoring Framework for Cloud Infrastructures. *IET Communications*, 6(10): 1306–1315, 2012. doi: 10.1049/iet-com.2011.0200. URL http://link.aip.org/link/?COM/6/1306/1.

[103] Dejan Kovachev, Yiwei Cao, and Ralf Klamma. Building mobile multimedia services: a hybrid cloud computing approach. *Multimedia Tools and Applications*, May 2012. ISSN 1380-7501. doi: 10.1007/s11042-012-1100-6. URL http://www.springerlink.com/index/10.1007/s11042-012-1100-6.

[104] H Krawczyk, M Bellare, and R Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), February 1997. URL http://www.ietf.org/rfc/rfc2104.txt.

[105] S Lai, J Cao, and Y Zheng. PSWare: A publish/subscribe middleware supporting composite event in wireless sensor network. *Pervasive Computing and ...*, 2009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4912862.

[106] Sean Lawlor, Patrick Diez, and Frank Ferrie. Real-Time Distributed Computing: CHOPS. In *Canadian Conference on Electrical and Computer Engineering*, Montreal, Quebec, 2012.

[107] P Leach, M Mealling, and R Salz. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122 (Proposed Standard), July 2005. URL http://www.ietf.org/rfc/rfc4122.txt.

[108] X Li. An Efficient Design of Publication and Subscription Model Based on WSN. *...on Information Technology and Management Science ...*, 2013. URL http://link.springer.com/chapter/10.1007/978-3-642-34910-2_52.

[109] X Li, MH Ammar, and S Paul. Video multicast over the Internet. *Network, IEEE*, 1999. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=768488.

[110] Xue Li, Sanjoy Paul, and Mostafa H Ammar. Layered video multicast with retransmission (LVMR): Evaluation of hierarchical rate control. *In Proc. NOSSDAV*, pages 1–26, 1998. URL http://eprints.kfupm.edu.sa/47925/.

[111] Xue Li, Bell Laboratories-lucent Technologies, Mostafa H Ammar, and Sanjoy Paul. Multicast over the Internet. *Ieee Network*, pages 46–60, 1999.

[112] Libav Project. Libav: Open source audio and video processing tools, 2013. URL http://libav.org.

[113] Y Liu, Y Guo, and C Liang. A survey on peer-to-peer video streaming systems. *Peer-to-peer Networking and Applications*, 2008. URL http://link.springer.com/article/10.1007/s12083-007-0006-y.

[114] J Lopez-Vega, Javier Povedano-Molina, Javier Sanchez-Monedero, and Juan Manuel Lopez-Soler. Políticas de QoS en una Plataforma de Trabajo Colaborativo sobre Middleware DDS. *XIII Jornadas de ...*, 2010. URL http://dtstc.ugr.es/tl/pdf/JTR2010/jmlvega_ptc.pdf.

[115] J.M. Lopez-Vega, J. Povedano-Molina, and Juan M. Lopez-Soler. QoS Policies for Audio/Video Distribution Over DDS Middleware. In *Real-time and Embedded Systems Workshop*, 2008.

[116] Matthew L Massie, Brent N Chun, and David E Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, July 2004. ISSN 01678191. doi: 10.1016/j.parco.2004.04.001. URL http://linkinghub.elsevier.com/retrieve/pii/S0167819104000535.

[117] M MATÝSEK, M ADÁMEK, M KUBALÄŒÍK, and M MIHOK. Monitoring of computer networks and applications using Nagios. *wseas.us*, 2012. URL http://www.wseas.us/e-library/conferences/2012/Sliema/DNMAT/DNMAT-07.pdf.

[118] P Millard, P Saint-Andre, and R Meijer. XEP-0060: Publish-subscribe. *Jabber Software Foundation*, 2006. URL http://xmpp.org/extensions/xep-0060.pdf.

[119] I Mironov. Hash functions: Theory, attacks, and applications. *Microsoft Research, Silicon Valley Campus. Noviembre ...*, 2005. URL http://131.107.65.14/en-us/people/mironov/hash_survey.pdf.

[120] J Moffitt. Ogg vorbis—Open, free audio—Set your media free. *Linux journal*, 2001. URL http://dl.acm.org/citation.cfm?id=364691.

[121] P Moghe. Adaptive polling rate algorithm for SNMP-based network monitoring. *US Patent 6,173,323*, 2001. URL http://www.google.com/patents?hl=en&lr=&vid=USPAT6173323&id=XW18AAAAEBAJ&oi=fnd&dq=Adaptive+polling+rate+algorithm+for+SNMP-based+network+monitoring&printsec=abstract.

[122] MQTT Community. MQ Telemetry Transport Website, 2011. URL http://mqtt.org/.

[123] MGS Nagaraja, RR Chittal, and K Kumar. Study of network performance monitoring tools-SNMP. *IJCSNS*, 2007. URL http://paper.ijcsns.org/07_book/200707/20070743.pdf.

[124] Nagios Enterprises LLC. Nagios, 2013. URL http://www.nagios.org.

[125] Nagios Enterprises LLC. Nagios Customers and Users, 2013. URL http://www.nagios.com/users/.

[126] Nagios Enterprises LLC. Nagios Exchange, 2013. URL http://exchange.nagios.org/directory/Plugins.

[127] Klara Nahrstedt and Wolf-Tilo Balke. A taxonomy for multimedia service composition. In *Proceedings of the 12th annual ACM international conference on Multimedia - MULTIMEDIA '04*, page 88, New York, New York, USA, 2004. ACM Press. ISBN 1581138938. doi: 10.1145/1027527.1027544. URL http://portal.acm.org/citation.cfm?doid=1027527.1027544.

[128] Teodor Neagoe, Valentin Cristea, and Logica Banica. NTP versus PTP in Com puter Networks Clock Synchronization. *Industrial Electronics, 2006 ...*, pages 0–5, 2006. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4077944.

[129] HB Newman, IC Legrand, and P Galvez. Monalisa: A distributed monitoring service architecture. *arXiv preprint cs/ ...*, 2003. URL http://arxiv.org/abs/cs/0306096.

[130] Oasis Consortium. OASIS, 2013. URL http://www.oasis-open.org.

[131] Object Management Group. CORBA 3.0 - General Inter-ORB Protocol chapter. Technical report, Object Managemen Group, 2002.

[132] Object Management Group. Data Distribution Service for Real-time Systems, 2007. URL http://www.omg.org/spec/DDS/1.2/.

[133] Object Management Group. The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification. *Management*, 15(January), 2009.

[134] Object Management Group. DDS Security (Request for Proposal). Technical report, Object Management Group, 2010. URL http://www.omg.org/cgi-bin/doc?mars/2010-12-37.

[135] Object Management Group. Extensible and Dynamic Topic Types for DDS. Technical Report May 2010, Object Management Group, 2011.

[136] OpenCV Project. OpenCV, 2013. URL http://opencv.org/.

[137] Oracle. OpenMQ, 2013. URL mq.java.net/.

[138] Z Ouyang, F Dai, J Guo, and Y Zhang. VTrans: A Distributed Video Transcoding Platform. *Advances in Multimedia Modeling*, 2013. URL http://link.springer.com/chapter/10.1007/978-3-642-35728-2_60.

[139] G. Pardo-Castellote. Omg data-distribution service: Architectural overview. *Distributed Computing Systems ...*, pages 200–206, 2003. doi: 10.1109/ICDCSW.2003.1203555. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1203555http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1203555.

[140] Michael Pasieka, Paul Crumley, Ann Marks, and Ann Infortuna. Distributed Multimedia: How Can the Necessary Data Rates be Supported? In *USENIX Summer*, pages 169–182. Carnegie Mellon University, 1991. URL http://reports-archive.adm.cs.cmu.edu/anon/anon/home/ftp/usr0/ftp/itc/CMU-ITC-107.pdf.

[141] Roy D. Pea and Louis M. Gomez. Distributed multimedia learning environments: Why and How. *Interactive Learning Environments*, 2(2), 1992. URL ftp://ftp.cs.ubc.ca/.snapshot/nightly.6/local/pstv/DMSbook/CalvertReady.doc.

[142] D PENG, L CAO, and W XU. Using JSON for Data Exchanging in Web Service Applications. *Journal of Computational Information Systems*, 2011. URL http://www.jofcis.com/publishedpapers/2011_7_16_5883_5890.pdf.

[143] V Phuah and J Diaz-Gonzalez. Developing distributed multimedia applications. *ACM SIGCOMM ...*, 1992. URL http://dl.acm.org/citation.cfm?id=142307.

[144] Planet Lab Project. Planet Lab, 2013. URL http://www.planet-lab.org.

[145] J. Povedano-Molina, Jose M. Lopez-Vega, and Juan M. Lopez-Soler. EMDS: an Extensible Multimedia Distribution Service. In *Real-time and Embedded Systems Workshop*, 2010.

[146] J Povedano-Molina, Jose M. Lopez-Vega, J.M. Lopez-Soler, Antonio Corradi, and Luca Foschini. DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant clouds. *Future Generation ...*, 2013. URL http://www.sciencedirect.com/science/article/pii/S0167739X13000824.

[147] Javier Povedano-Molina, Jose M. Lopez-Vega, Javier Sanchez-Monedero, and Juan M. Lopez-Soler. Instant Messaging Based Interface for Data Distribution Service. In *XIII Jornadas de Tiempo Real*, 2010.

[148] L Prechelt. An empirical comparison of seven programming languages. *Computer*, 2000. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=876288.

[149] PSIRP project. PSIRP, 2013. URL http://www.psirp.org/.

[150] PURSUIT Project. PURSUIT, 2013. URL http://www.fp7-pursuit.eu/PursuitWeb/.

[151] B Rajagopalan. Electricity over IP. RFC 3251 (Informational), April 2002. URL http://www.ietf.org/rfc/rfc3251.txt.

[152] Real-time Innovations Inc. Meeting Real-Time Requirements in Integrated Defense Systems. Technical report, Real-Time Innovations Inc., 2007. URL http://www.rti.com/whitepapers/Real-Time_Integrated_Defense_Systems.pdf.

[153] Real-time Innovations Inc. RTI in Financial Services: Performance, Consistency, Reliability. Technical report, Real Time Innovations Inc., 2007. URL http://www.rti.com/whitepapers/RTI_Finance_WP.pdf.

[154] R Van Renesse, KP Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer ...*, 21(2):164–206, 2003. URL http://dl.acm.org/citation.cfm?id=762485.

[155] E Rescorla and N Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), April 2006. URL http://www.ietf.org/rfc/rfc4347.txt.

[156] E Rescorla and N Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347 (Proposed Standard), January 2012. URL http://www.ietf.org/rfc/rfc6347.txt.

[157] RESERVOIR FP7 Project. RESERVOIR: Resources and Services Virtualization without Barriers, 2013. URL http://www.reservoir-fp7.eu/.

[158] M Ripeanu. Peer-to-peer architecture case study: Gnutella network. *Peer-to-Peer Computing, 2001. Proceedings. First ...*, 2001. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=990433.

[159] C Rodrigues, J Afonso, and P Tomé. Mobile Application Webservice Performance Analysis: Restful Services with JSON and XML. *ENTERprise Information Systems*, 2011. URL http://link.springer.com/chapter/10.1007/978-3-642-24355-4_17.

[160] P Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920 (Proposed Standard), October 2004. URL http://www.ietf.org/rfc/rfc3920.txt.

[161] P Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. RFC 3921 (Proposed Standard), October 2004. URL http://www.ietf.org/rfc/rfc3921.txt.

[162] J. Sanchez-Monedero, J. Povedano-Molina, J.M. Lopez-Vega, and J.M. Lopez-Soler. Bloom filter based discovery protocol for DDS middleware. *Journal of Parallel and Distributed Computing*, 2011. URL http://www.citeulike.org/group/14279/article/9325093.

[163] D Schreiber and M Muhlhauser. Configurable Middleware for Multimedia Collaboration Applications. *Multimedia (ISM), ...*, 2012. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6424681.

[164] AT Schreiner. *Object oriented programming with ANSI-C*. Schreiner, AT, 1993. URL https://ritdml.rit.edu/handle/1850/8544.

[165] H Schulzrinne, S Casner, R Frederick, and V Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. URL http://www.ietf.org/rfc/rfc3550.txt.

[166] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007. ISSN 1051-8215. doi: 10.1109/TCSVT.2007.905532. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4317636.

[167] Securities Technology Analysis Center LLC. STAC-MC2, 2013. URL http://www.stacresearch.com/m2.

[168] K Shi, Z Deng, and X Qin. Tinymq: A content-based publish/subscribe middleware for wireless sensor networks. ..., *The Fifth International Conference on Sensor ...*, 2011. URL http://www.thinkmind.org/index.php?view=article&articleid=sensorcomm_2011_1_30_10072.

[169] L Skorin-Kapov. Application-level QoS negotiation and signaling for advanced multimedia services in the IMS. *Communications ...*, 2007. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4268292.

[170] M Srivatsa and L Liu. Secure event dissemination in publish-subscribe networks. ... *Computing Systems, 2007. ICDCS'07. 27th ...*, 2007. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4268177.

[171] NF Standard. Announcing the Advanced Encryption Standard (AES). *Federal Information Processing Standards Publication*, 2001. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Announcing+the+ADVANCED+ENCRYPTION+STANDARD+(AES)#0.

[172] Mark Stefik, Gregg Foster, and DG Bobrow. Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Communications of the ...*, 30(1), 1987. URL http://dl.acm.org/citation.cfm?id=7887.

[173] I Stoica, R Morris, and D Karger. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM ...*, 2001. URL http://dl.acm.org/citation.cfm?id=383071.

[174] R Subramanyan. A scalable SNMP-based distibuted monitoring system for heterogeneous network computing. *Proceedings of the 2000 ...*, 2000. URL http://dl.acm.org/citation.cfm?id=370073.

[175] GA Thom. H. 323: the multimedia communications standard for local area networks. *Communications Magazine, IEEE*, 1996. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=556487.

[176] TIBCO. Rendezvous White Paper. Technical report, TIBCO, 1999. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Rendezvous+White+Paper#0.

[177] Thanassis Tiropanis and Dimitris Kanellopoulos. A schema-based P2P network to enable publish-subscribe for multimedia content in open hypermedia systems. *International Journal of Web Engineering and Technology*, 4(1):21, 2008. ISSN 1476-1289. doi: 10.1504/IJWET.2008.016103. URL http://www.inderscience.com/link.php?id=16103.

[178] C Topolcic. Experimental Internet Stream Protocol: Version 2 (ST-II). RFC 1190 (Experimental), October 1990. URL http://www.ietf.org/rfc/rfc1190.txt.

[179] S Turner and L Chen. Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms. RFC 6151 (Informational), March 2011. URL http://www.ietf.org/rfc/rfc6151.txt.

[180] Ahmet Uyar, S Pallickara, and G Fox. Towards an Architecture for Audio/Video Conferencing in Distributed Brokering Systems. *Proceedings of the 2003 ...*, 2003. URL http://grids.ucs.indiana.edu/users/shrideep/narada/papers/NB-AudioVideo.pdf.

[181] A Vakili and JC Grégoire. QoE Management for Video Conferencing Applications. *Computer Networks*, 2013. URL http://www.sciencedirect.com/science/article/pii/S1389128613000558.

[182] VCL VCL. H.264/AVC Video Coding Standard. Technical Report April, ITU-T, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.1310&rep=rep1&type=pdf.

[183] R Verhoeven and J Lukkien. A publish-subscribe Architecture for Interoperable in-home Video Streaming. In *ECRTS Workshop on Real-time for Multimedia*, pages 13–19, Eindhoven, 2004. URL http://alexandria.tue.nl/extra1/wskrap/publichtml/200436.pdf#page=11.

[184] S Vinoski. Advanced message queuing protocol. *Internet Computing, IEEE*, 2006. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4012603.

[185] S Vinoski. Demystifying restful data coupling. *Internet Computing, IEEE*, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4463391.

[186] VMWare Inc. VMWare Inc., 2013. URL http://www.vmware.com/.

[187] VMWare Inc. Hyperic: Systems Monitoring, Server Monitoring & Systems Management Software, 2013. URL http://www.hyperic.com.

[188] Werner Vogels. Beyond Server Consolidation. *Queue*, 6(February):20–26, 2008.

[189] S Waldbusser, R Cole, C Kalbfleisch, and D Romascanu. Introduction to the Remote Monitoring (RMON) Family of MIB Modules. RFC 3577 (Informational), August 2003. URL http://www.ietf.org/rfc/rfc3577.txt.

[190] N Williams, GS Blair, and N Davies. Distributed multimedia computing: an assessment of the state of the art. *Information Services and Use*, 1(11), 1991. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.2785&rep=rep1&type=pdf.

[191] Wireshark Foundation. Wireshark, 2013. URL http://www.wireshark.org.

[192] JL Wolf, PS Yu, and H Shachnai. Disk load balancing for video-on-demand systems. *Multimedia Systems*, 1997. URL http://www.springerlink.com/index/5hd0kdrf8u0q67nt.pdf.

[193] S Wu, W Zhang, and H Wang. PS4WSN-a publish/subscribe middleware for wireless sensor networks. *Information and Automation (ICIA), ...*, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5512444.

[194] T Ylonen and C Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard), January 2006. URL http://www.ietf.org/rfc/rfc4251.txt.

[195] Hang Yuan, C.C.J. Kuo, and I. Ahmad. Energy efficiency in data centers and cloud-based multimedia services: An overview and future directions. In *Green Computing Conference, 2010 International*, pages 375–382. IEEE, 2010. ISBN 9781424476145. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5598292.

[196] C.-H.P. Yuen and S.-H.G. Chan. Scalable Real-Time Monitoring for Distributed Applications. *Parallel and Distributed Systems, IEEE Transactions on*, 23(12):2330–2337, 2012. ISSN 1045-9219. doi: 10.1109/TPDS.2012.60. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6148226.

[197] W Yun. Realization of a Multimedia Conferencing System Based on Campus Network. *Instrumentation, Measurement, Circuits and Systems*, 2012. URL http://link.springer.com/chapter/10.1007/978-3-642-27334-6_73.

[198] Zenoss Inc. Zenoss, 2013. URL http://www.zenoss.com/.

[199] L Zhai, C Li, and L Sun. Research on the Message-Oriented Middleware for Wireless Sensor Networks. *Journal of Computers*, 2011. URL https://academypublisher.com/~academz3/ojs/index.php/jcp/article/view/060510401046.

[200] W Zhu, C Luo, J Wang, and S Li. Multimedia cloud computing. *Signal Processing Magazine, ...*, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5754008.

[201] Y Zou, J Cao, and H Wu. TraffiCast: Real-time Pub/Sub based video surveillance system over interconnected WMNs and WSNs. *Distributed Computing in Sensor Systems ...*, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5593283.

[202] Yang Zou. Pub-Eye: The delay constrained Pub/Sub for large scale wireless video surveillance. *Wireless Communications and ...*, pages 2149–2154, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5779465.