

OFFICIAL GRADUATE PROGRAM IN MULTIMEDIA SYSTEMS

Signal Theory, Telematics and Communications Department

UNIVERSITY OF GRANADA



Ph.D. Thesis Dissertation:

**Discovery and Signaling
Enhancements
for Data-Centric Publish-Subscribe
Environments**

Written by:

Jose Maria Lopez-Vega

Supervised by:

PhD. Juan Manuel Lopez-Soler

PhD. Gerardo Pardo-Castellote

Editor: Editorial de la Universidad de Granada
Autor: José María López Vega
D.L.: GR 95-2014
ISBN: 978-84-9028-671-5

PROGRAMA OFICIAL DE POSGRADO EN SISTEMAS MULTIMEDIA

Departamento de Teoría de la Señal, Telemática y Comunicaciones

UNIVERSIDAD DE GRANADA



TESIS DOCTORAL

**Mejoras de Descubrimiento
y Señalización para Entornos
Publicación-Suscripción
Centrada en Datos**

Realizada por:

José María López Vega

Dirigida por:

Dr. Juan Manuel López Soler

Dr. Gerardo Pardo Castellote

El doctorando D. José María López Vega y los directores de la tesis D. Juan Manuel López Soler, Profesor Titular de Universidad del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada, y D. Gerardo Pardo Castellote, CTO. de Real-Time Innovations Inc.

GARANTIZAMOS AL FIRMAR ESTA TESIS DOCTORAL

que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro 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 08 de abril de 2013

Directores de la Tesis

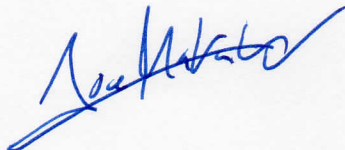


D. Juan Manuel López Soler



D. Gerardo Pardo Castellote

Doctorando



D. José María López Vega

Agradecimientos

En primer lugar, querría agradecer a mis directores, Juanma y Gerardo, por su labor de supervisión de este trabajo, así como por su continuo apoyo. En especial, agradecerle a Juanma que durante estos 6 años no sólo ha sido un director, sino un compañero de trabajo y amigo.

En segundo lugar, a mi grupo de investigación, Wireless and Multimedia Networking Lab (WMNL), cuyos miembros han sido compañeros de batallas en pos de la concesión de proyecto definitiva. En especial, agradecerles a Pove y Juanjo el buen humor con el que se enfrentan a *deadlines* y *cruasanes*.

Asimismo quiero agradecer al Departamento de Teoría de la Señal, Telemática, y Comunicaciones por haberme acogido como un miembro más y por haberme ayudado en todo lo que he necesitado.

Estoy muy agradecido a todos aquellos que han contribuido a este trabajo con sus ideas y sugerencias. En especial, a Javier Sanchez-Monedero por sus estupendas figuras de SDP y a Luca Foschini por sus valiosos comentarios, que han contribuido a hacer de éste un mejor documento.

No me puedo olvidar de mis compañeros la *comunidad del becario*, que se adaptan igual de bien a unas mazmorras sin ventanas que al frío de más allá del Mur... digo, orquídea. Y es que cuatro años de becario dan para mucho: tres mudanzas de despacho, algún pequeño incendio, dos o tres reformas, etc.. Por todo este tiempo juntos, muchas gracias a mis compis de despacho (Alberto, Didi, Lluvia, Rita, Rosa, Santi), de despacho anexo (Joaquín, Leopomodoro-breaker, Miriam, Plata, Roberto), de mazmorras (Fran, Juanra, Víctor), de máster (Miguel y Pablo), de almuerzo (Ruben) y bitstamineros (Fergu y Javi). Y muchas, muchas, muchas gracias adelantadas Carlos y Rafa por todos los ensayos de la defensa de esta Tesis a los que vais a asistir.

A toda la gente que me ayudó en mi periplo por *las dos estancias*. Agradecer a RTI, y en especial a Álex, Fernando y Gerardo, su confianza y apoyo durante mi estancia en EE.UU.. A Gonzalo, por brindarme la oportunidad de la estancia en Finlandia, que fue una experiencia increíble. Y no puedo olvidarme de Jaime y Ella, que fueron un apoyo constante y son unos verdaderos amigos.

A mis amigos del mundo exterior (a la investigación), que me han ayudado a llevar este *retorno del deadline* adelante: unos, con su paciencia infinita (Dewidh), otros, con un poco de cultura oriental (Bea, pinxaja, anubis, Juanma), y otros, con un toque musical (Goriwiiii & Master Baena).

Como no, agradecer a mi familia su apoyo continuo, si hoy puedo escribir estas líneas es gracias a ellos.

Y por último, pero no menos importante, a Adela, por ser mi compañera en este viaje que es la vida.

¡GRACIAS A TODOS!

Este trabajo ha sido parcialmente financiado por el programa "Formación de Profesorado Universitario 2008" del Ministerio de Educación".

Abstract

Current uses of the Internet are mostly built on request-response exchanges between pairs of computers. This model has been very successful at providing shared remote access to resources and point-to-point communications for applications including email, remote file and database access, web access, etc. The advent of ubiquitous Internet connectivity and cheap network-connected devices has greatly expanded the communication needs of applications using the Internet. For these new kinds of applications the request-response communications model, which is highly coupled to the location of the information, is cumbersome and limited whereas other communication models, such as publish-subscribe, provide a more natural fit.

In contrast to the request-response interaction model, in which a client sends a request and a server returns a response, in the publish-subscribe model information producers (publishers) are decoupled from information consumers (subscribers) via the publish-subscribe service. One of these services is Data Distribution Service (DDS), an standardized solution for data-centric publish-subscribe data distribution.

This Thesis describes the design and evaluation of a set of discovery and signaling enhancements for Data-Centric Publish-Subscribe (DCPS) systems. In particular, this Thesis focus on solving the following open issues of DDS: data-spaces interconnection, data transformation, Quality of Service (QoS) adaptation, universal DCPS entity and data-content naming, end-to-end session establishment, and discovery scalability in large deployments.

We have divided our solution in three parts.

First, we propose the DDS data-space Interconnection Service (DDS-IS), a fully DDS-compliant service for DDS data-spaces interconnection which also provides transparent data transformation and QoS adaptation. We demonstrate that the DDS-IS can improve the scalability of DDS systems by reducing the network traffic load between the interconnected data-spaces.

Secondly, we define a mechanism for universal identification of DCPS entities and data-content, and we propose the rationale and design of a protocol for session signaling in DCPS environments. This protocol allows applications to perform DCPS entity and data-content discovery in medium-scale environments. It also allows the creation of sessions between nodes, which eases the creation of DCPS routes and enables the enforcement of QoS at end-to-end level.

Finally, we propose a scalable Peer-to-Peer (P2P) solution for performing content discovery and data transfer. This solution is based on the REsource LOcation And Discovery (RELOAD) framework, one of the latest standards of the Internet Engineering Task Force (IETF). This solution provides rendezvous function for locating sensors, actuators, and applications that share a common data-space or topic

of interest using a DCPS approach. Once the rendezvous is complete, nodes use existing RELOAD features for exchanging information. We demonstrate the proposed solution scales well for large deployments and it is robust against node failures.

Resumen

En la actualidad, las aplicaciones distribuidas en Internet interactúan e intercambian información usando mayoritariamente relaciones del tipo solicitud-respuesta. Este modelo de interacción ha demostrado ser adecuado para comunicaciones punto a punto y más concretamente, para aplicaciones tales como el correo electrónico, la descarga de archivos, el acceso a bases de datos y los servicios web, entre otras. Sin embargo, la proliferación de múltiples dispositivos dotados de conectividad permanente a Internet y la aparición de nuevas aplicaciones, con diferentes necesidades, han motivado la consideración de otros modelos de interacción, ya que no siempre el modelo solicitud-respuesta tiene que ser el más adecuado. En este sentido, más recientemente han surgido otros modelos de interacción alternativos, como el de publicación-suscripción.

A diferencia del modelo solicitud-respuesta, en el que un cliente envía una solicitud, y un servidor devuelve una respuesta, en el modelo publicación-suscripción los productores de información (publicadores) están desacoplados de los consumidores de información (suscriptores) mediante un servicio publicación-suscripción. Uno de estos servicios es Data Distribution Service (DDS), una solución estandarizada para la distribución de datos mediante un paradigma publicación-suscripción centrada en datos.

Esta Tesis describe el diseño y evaluación de un conjunto de mejoras de descubrimiento y señalización para sistemas Data-Centric Publish-Subscribe (DCPS). En concreto, esta Tesis se centra en resolver los siguientes problemas de DDS: interconexión de dominios, transformación de datos, adaptación de Quality of Service (QoS), identificación universal de contenidos y entidades DCPS, establecimiento de sesión extremo a extremo y escalabilidad del descubrimiento en sistemas de gran escala.

Nuestra solución ha sido dividida en tres partes.

En primer lugar, se propone el DDS data-space Interconnection Service (DDS-IS), un servicio para la interconexión de dominios DDS que soporta la transformación transparente de datos y la adaptación de QoS, todo ello manteniendo la compatibilidad con el estándar. Además, se demuestra experimentalmente que el DDS-IS puede mejorar la escalabilidad de los sistemas DDS mediante la reducción del tráfico de red existente entre los dominios interconectados.

En segundo lugar, se define un mecanismo para la identificación universal de contenidos y entidades DCPS, y se propone el diseño y lógica subyacente de un protocolo para la señalización de sesión en entornos DCPS. Este protocolo permite a las aplicaciones descubrir contenidos y entidades DCPS en entornos de escala media. Además, permite la creación de sesiones entre nodos, lo que facilita la creación de rutas DCPS y permite la provisión de QoS extremo a extremo.

Por último, se propone una solución Peer-to-Peer (P2P) escalable para el descu-

brimiento de contenidos y la transferencia de información. Esta solución se basa en el *framework* REsource LOcation And Discovery (RELOAD), uno de los últimos estándares de la Internet Engineering Task Force (IETF). La solución propuesta proporciona la función de *rendezvous* para localizar mediante una aproximación DCPS sensores, actuadores y aplicaciones que comparten un espacio de datos común o un tópico de interés. Una vez se completa el *rendezvous*, los nodos utilizan la funcionalidad de RELOAD para intercambiar información. Por último, se demuestra experimentalmente que la solución propuesta es escalable y robusta.

Contents

1	Introducción	1
1.1	Motivación	3
1.1.1	Interconexión de dominios	3
1.1.2	Señalización	4
1.2	Objetivos	6
1.3	Resumen de la solución propuesta	7
1.4	Estado del arte y trabajo relacionado	7
1.4.1	Modelo de interacción publicación-suscripción	8
1.4.2	Escalabilidad de los sistemas publicación-suscripción	8
1.4.3	Descubrimiento de recursos y RELOAD	11
1.4.4	DDS y SIP	12
1.5	Contribuciones principales	12
1.5.1	Trabajos publicados	13
1.6	Estructura del documento	14
1	Introduction	17
1.1	Motivation	18
1.1.1	Domain interconnection	19
1.1.2	Signaling	20

1.2	Goals	21
1.3	Solution overview	22
1.4	State of the art and related work	23
1.4.1	Publish-subscribe interaction model	23
1.4.2	Scalability in publish-subscribe systems	24
1.4.3	Resource discovery and RELOAD	25
1.4.4	DDS and SIP	26
1.5	Main contributions	27
1.5.1	Published works	27
1.6	Document structure	29
2	Background	31
2.1	Publish-subscribe interaction model	31
2.2	Data Distribution Service (DDS)	33
2.2.1	Data Centric Publish Subscribe layer (DCPS)	34
2.2.2	Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol (DDS-RTPS)	36
2.2.3	SDP Bloom	39
2.2.4	Extensible Topic Types	40
2.3	Session Initiation Protocol (SIP)	41
2.3.1	Overview of SIP functionality	41
2.3.2	Basic concepts of SIP	42
2.3.3	User location	43
2.3.4	Session setup and management	45
2.4	REsource LOcation And Discovery (RELOAD)	47
2.4.1	Main features	47
2.4.2	Architecture and types of nodes	48
2.4.3	Operations	49

2.4.4	Data storage	50
2.4.5	Usages	51
2.5	Constrained Application Protocol (CoAP)	51
3	DDS data-space Interconnection Service (DDS-IS)	53
3.1	Motivation	53
3.2	Motivating use case example	55
3.3	Requirements	57
3.4	Internal architecture	58
3.4.1	Application layer	59
3.4.2	Routing layer	60
3.4.3	Pub/Sub and network layers	62
3.5	System operation	63
3.5.1	Discovery of new DDS entities	64
3.5.2	Data bridging	65
3.6	Implementation	67
3.6.1	Dynamic types	67
3.6.2	DDS middleware implementation	67
3.6.3	DDS signaling propagation	67
3.6.4	Propagating DataWriter information	68
3.6.5	Quality of Service	69
3.6.6	XML configuration format	69
3.7	Validation and performance evaluation	71
3.7.1	Experimental set-up	71
3.7.2	DDS-IS Impact in N to N communications	74
3.7.2.1	Performance impact in simple 1 to 1 communications	75
3.7.2.2	Performance impact of data transformation and filtering features	76

3.7.2.3	Performance impact in N to N communications with multiple topics	77
3.7.3	DDS-IS impact in M to N communications	80
3.7.3.1	DDS-IS performance impact	81
3.7.3.2	DDS-IS scalability impact	83
3.8	Impact on DDS QoS policies	85
3.9	Conclusions	92
4	Data-Centric SIP (DCSIP)	95
4.1	Motivation	95
4.2	Motivating use case example	96
4.3	New required features	100
4.3.1	Limitations of DDS and DDS-IS	100
4.3.2	Solution requirements	102
4.4	Design decisions	103
4.4.1	Information management	103
4.4.2	Signaling	104
4.4.3	Routing	105
4.5	DCSIP fundamentals	106
4.5.1	Realm	107
4.5.2	Scope, scope-label, and scope-path	107
4.5.3	Full Qualified Data-space and Topic Names	107
4.5.4	DCPS Entity Group and DCPS Entity Name	108
4.5.5	DCPS session	108
4.6	Resource naming format	109
4.7	Node roles	110
4.8	DCSIP messages	111
4.8.1	Joining and registration	111

4.8.2	Resource location	112
4.8.3	Session establishment and maintenance	112
4.9	DCSIP operation	113
4.9.1	Joining and registration	113
4.9.2	Resource location and session establishment	115
4.10	SIP and DCSIP comparison	117
4.11	DDS and DCSIP features comparison	126
4.12	Conclusions	128
5	DDS usage for RELOAD	129
5.1	Motivation	129
5.2	Motivating use case examples	131
5.2.1	User data-space	131
5.2.2	N permanent stock-shares publishers, M variable stock-shares subscribers	132
5.2.3	N variable temperature sensors, 1 permanent temperature monitor	132
5.2.4	N variable planes (entering and leaving the airspace), M variable air traffic controllers	132
5.3	New required features	133
5.4	System design	133
5.4.1	A DCPS approach	133
5.4.2	A RELOAD based architecture	134
5.4.3	Discovery-and-Data node roles	136
5.4.4	DCPS Entity Groups	136
5.4.5	RELOAD Kinds for DCPS	136
5.4.5.1	TopicPublisher and TopicSubscriber Registrations	137
5.4.5.2	DataSpaceParticipant Registration	138
5.4.6	Resource naming format	139

5.5	System operation	140
5.5.1	Joining the overlay	140
5.5.2	Registering entities	141
5.5.2.1	Registering a new publisher using TopicPublisher . .	141
5.5.2.2	Registering a new subscriber using TopicSubscriber .	142
5.5.2.3	Registering a new data-participant using DataSpaceParticipant	143
5.5.3	Entity Discovery	144
5.5.3.1	Publisher discovery	144
5.5.3.2	Subscriber discovery	145
5.5.3.3	Data-participant discovery	145
5.5.4	Data transfer	146
5.5.4.1	Data transfer using CoAP	147
5.5.4.2	Data transfer using DDS-RTPS	147
5.6	Prototype and experimental setup	147
5.6.1	Experimental setup	148
5.7	Experimental results	150
5.8	Conclusions	153
6	Related Work	155
6.1	System federation and domain bridging in DDS	155
6.2	DDS and SIP	157
6.3	RELOAD and publish-subscribe scalability	157
6.4	CoAP usage for RELOAD	158
6.5	Named Data Networking (NDN)	158
6.6	Conclusions	159
7	Conclusions and future work	161

7.1	Conclusions	161
7.2	Future work	165
7	Conclusiones y trabajo futuro	167
7.1	Conclusiones	167
7.2	Trabajo futuro	171
	Bibliography	173

List of Figures

2.1	Request-response and publish-subscribe interaction models.	32
2.2	DDS concepts and entities.	35
2.3	DCPS built-in entities for discovery purposes.	36
2.4	SDP sequence diagram.	38
2.5	SDP-Bloom sequence diagram.	39
2.6	SIP registration and invitation example.	45
2.7	SIP invitation dialog.	46
2.8	Trivial example of RELOAD overlay.	49
2.9	Example of data stored in RELOAD.	50
3.1	Example deployment scenario.	56
3.2	Layered DDS-IS architecture.	59
3.3	DDS-IS Routing Engine.	61
3.4	Discovery sequence diagram.	64
3.5	Data bridging sequence diagram.	66
3.6	XML configuration file example.	70
3.7	DDS-IS scenario for 1 to 1 performance experiments. In the N to N case, lab01 and lab03 run multiple publishers and subscribers.	74
3.8	DDS-IS impact for different data types in 1 to 1 communications.	75

3.9	DDS-IS performance for filtering and transforming data.	76
3.10	Average latency for different payloads and number of routes (topics) active.	79
3.11	Total average throughput for different payloads and number of routes (topics) active.	79
3.12	CPU impact for different payloads and number of routes (topics) active.	80
3.13	System topology used in 1 to N performance experiments.	81
3.14	Average latency for different payloads when demultiplexing data from 1 to N	82
3.15	Throughput per subscriber for different payloads when demultiplexing data from 1 to N	82
3.16	System topology used in M to N scalability experiments.	83
3.17	DDS traffic load on LAN C for both no-DDS-IS and DDS-IS scenarios.	85
4.1	Topology of the Health Service use-case.	97
4.2	DDS-IS are able to integrate incompatible information data-bases. . .	98
4.3	Current solutions provoke unnecessary and redundant traffic in the higher levels of the DDS-IS hierarchy.	99
4.4	Our solution allows for a better usage of the resources.	101
4.5	Node joining and content registering.	114
4.6	Session creation diagram.	116
5.1	Architecture layers.	135
5.2	Example of overlay prior to any DataRegister.	141
5.3	Example of overlay after several DataRegister.	144
5.4	A) Average latency for storing DCPS entities discovery information. B) Average latency for obtaining all the discovery information of a particular DCPS-Entity-Group.	151
5.5	Average latency for obtaining the discovery information for a particular DCPS-Entity-Group and start exchanging data with a node of that group.	152

5.6 Percentages of lookup failures per different churn levels. 153

List of Tables

2.1	Correspondence DDS and DDS-RTPS entities.	37
3.1	Study of the level impact of DDS-IS on DDS QoS policies.	86
4.1	Comparison of SIP and DCSIP features.	117
4.2	Comparison of current DDS and DCSIP features.	126

List of Abbreviations and Acronyms

AMQP Advanced Message Queuing Protocol

API Application Programming Interface

ATC Air Traffic Controller

B-DR Built-in DataReader

B-DW Built-in DataWriter

CDN Content Distribution Network

CoAP Constrained Application Protocol

CoRE WG Constrained RESTful Environments working group

CPU Central Processing Unit

DCPS Data-Centric Publish-Subscribe

DCSIP Data-Centric SIP

DDS Data Distribution Service

DDS-IS DDS data-space Interconnection Service

DDS-RTPS Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol

DDS-XTypes Extensible And Dynamic Topic Types For DDS

DHT Distributed Hash Table

DLRL Data Local Reconstruction Layer

DNS	Domain Name System
DR	DataReader
DW	DataWriter
EDP	Endpoint Discovery Protocol
ESB	Enterprise Service Bus
FIR	Flight Information Region
FQDN	Full Qualified Domain Name
FQTN	Full Qualified Topic Name
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICE	Interactive Connectivity Establishment
ID	IDentifier
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
JMS	Java Messaging Service
LAN	Local Area Network
M2M	Machine-to-Machine
MANET	Mobile ad hoc network
MMUSIC WG	Multiparty MUltimedia SessIon Control working group
MOM	Message Oriented Middleware
NAT	Network Address Translation
NDN	Named Data Networking
NTP	Network Time Protocol
OMG	Object Management Group

-
- P2P** Peer-to-Peer
- P2PSIP WG** Peer-to-Peer Session Initiation Protocol working group
- P2P-SIP** Peer-to-Peer Session Initiation Protocol
- PDP** Participant Discovery Protocol
- PSIRP** Publish-Subscribe Internet Routing Paradigm
- PURSUIT** Publish Subscribe Internet Technology
- QoS** Quality of Service
- RELOAD** REsource LOcation And Discovery
- REVENGE** REliable and VErsatile News delivery support for aGencies
- RFC** Request For Comments
- RTI** Real Time Innovations Inc.
- RTP** Real-time Transport Protocol
- RTPS** Real-Time Publish-Subscribe
- SDP** Simple Discovery Protocol
- sdp** Session Description Protocol
- SEDP** Simple Endpoint Discovery Protocol
- SPDP** Simple Participant Discovery Protocol
- SIP** Session Initiation Protocol
- SIP WG** Session Initiation Protocol working group
- SIPCore WG** Session Initiation Protocol Core working group
- SIPPING WG** Session Initiation Protocol Project INvestiGation working group
- SNMP** Simple Network Management Protocol
- SPHS** Spanish Public Health Service
- TCP** Transmission Control Protocol
- TLS** Transport Layer Security
- UA** User Agent

UAC User Agent Client

UAS User Agent Server

UDP User Datagram Protocol

URI Uniform Resource Identifier

VLAN Virtual Local Area Network

WAN Wide Area Network

XML eXtensible Markup Language

Introducción

EN la actualidad, las aplicaciones distribuidas en Internet interactúan e intercambian información usando mayoritariamente relaciones del tipo solicitud-respuesta. Este modelo de interacción ha demostrado ser adecuado para comunicaciones punto a punto y más concretamente, para aplicaciones tales como el correo electrónico, la descarga de archivos, el acceso a bases de datos y los servicios web, entre otras.

La arquitectura de Internet, caracterizada por la sencillez y eficacia de su diseño, se concibió para facilitar la interconexión de redes –independientemente de las tecnologías subyacentes– y para permitir así el intercambio de información entre las aplicaciones finales, como las anteriormente citadas.

Para proporcionar comunicaciones fiables (sin errores), con control de flujo y con mecanismos de control de congestión, se especificó el protocolo de transporte Transmission Control Protocol (TCP), el cual provee estos servicios exclusivamente para comunicaciones punto a punto. TCP adopta una aproximación orientada a conexión (*full-duplex*) que implica un fuerte acoplamiento temporal y espacial entre el emisor y el receptor de la información.

La proliferación de múltiples dispositivos dotados de conectividad permanente a Internet y la aparición de nuevas aplicaciones, con diferentes necesidades, han motivado la consideración de otros modelos de interacción, ya que no siempre el modelo solicitud-respuesta tiene que ser el más adecuado. En este sentido, más recientemente han surgido otros modelos de interacción alternativos, como el de publicación-suscripción [21].

La naturaleza y el contexto histórico de la cuestión aquí planteada quedan claramente expresados en el trabajo pionero de Van Jacobson *et al.* [41], que traducido al español dice:

Los principios de ingeniería y la arquitectura de Internet –tal y como la conocemos hoy– fueron creados en los años 60 y 70, cuando la compartición de recursos era el problema que las redes de ordenadores pretendían resolver [...]. El modelo de comunicación que se concibió entonces consistía en una conversación entre dos máquinas, una solicitando un recurso y otra facilitándolo. Consecuentemente, los paquetes IP contienen dos identificadores (direcciones), uno para el equipo origen y otro para el equipo destino, y casi todo el tráfico de Internet consiste en conversaciones (TCP) entre pares de equipos.

*En los 50 años que han pasado desde la creación de las primeras redes de paquetes, los ordenadores y otros dispositivos relacionados se han abaratado y se han convertido en productos de uso cotidiano. La conectividad que ofrece Internet y el coste reducido del almacenamiento permiten el acceso a una cantidad de contenido asombrosa – solo en 2008 se han generado 500 exabytes de información [24]. La gente valora Internet por **qué** contenido le ofrece, no por **dónde** se encuentra dicho contenido.*

Hemos identificado una serie de problemas que afectan a los usuarios y cuyo origen se encuentra en la incompatibilidad entre estos dos modelos:

Disponibilidad: El acceso rápido y fiable al contenido requiere mecanismos específicos de aplicación incómodos y preplanificados como pueden ser las redes CDNs y P2P, así como imponer costes de ancho de banda excesivos.

Seguridad: Dado que los equipos confían en la información de localización y conexión para identificar los contenidos, es fácil suplantar dichos contenidos.

Dependencia espacial: Asociar contenido a la localización de los equipos complica la configuración y la implementación de servicios en red. La forma directa y unificada de resolver estos problemas es reemplazar el dónde con el qué. Las conversaciones equipo a equipo son una abstracción que se ajustaba a los problemas que existían en los años 60. Creemos que para los problemas de comunicación de actuales nombrar los datos es una mejor abstracción que nombrar los equipos.

A diferencia del modelo solicitud-respuesta, en el que la entidad denominada cliente envía una solicitud y la entidad denominada servidor le devuelve la respuesta, esta Tesis –en este sentido alineada con la visión de Jacobson– se centra en el modelo de interacción publicación-suscripción. En este modelo, unas entidades denominadas publicadores generan eventos o actualizan un espacio de datos compartido, mientras que otras, los suscriptores expresan su interés en ciertos patrones de eventos o partes del espacio de datos. Además, un servicio de publicación-suscripción se encarga de entregar la información desde los publicadores a los suscriptores.

Esta misma visión es igualmente compartida por varios proyectos de investigación recientes –encontrados en la denominada Internet del futuro– cuyo objetivo es sustituir al actual modelo de Internet adoptando el paradigma de publicación-suscripción. En concreto, por su relevancia son dignos de mención los proyectos Publish-Subscribe Internet Routing Paradigm (PSIRP) [20] [22] y Publish Subscribe Internet Technology (PURSUIT) [82].

1.1 Motivación

En 2004, el Object Management Group (OMG) [73] adoptó Data Distribution Service (DDS) [67], una especificación que estandariza el modelo de comunicación publicación-suscripción centrado en datos (DCPS). Esta especificación define una Interfaz de Programación de Aplicación (API) estandarizada para intercambiar información siguiendo una aproximación DCPS. Bajo este modelo, los publicadores y suscriptores intercambian datos a través de una caché distribuida conocida como espacio de datos (también conocida como dominio DDS). Otra característica significativa de DCPS es que los datos intercambiados no son ajenos al *middleware*. En este sentido, DDS es capaz de proporcionar funciones avanzadas tales como *caching* de los datos o filtrado basado en contenido.

El estándar DDS está todavía en expansión. En particular, y como prueba de la actividad existente en el estándar, en los últimos dos años el OMG ha publicado cuatro de los seis documentos que conforman la familia de estándares DDS. Una consecuencia de la relativa novedad de DDS es que –como se explicará más adelante– aún quedan varios problemas por resolver.

DDS fue originalmente concebido para su uso en redes de área local (LANs) aisladas, de tal forma que publicadores y suscriptores comparten una infraestructura de red común.

Si un suscriptor desea acceder a datos que se originan en un espacio de datos diferente, especialmente si dicho espacio de datos está separado por una red en desventaja, si hay implícito un servicio de traducción de nombres (NAT) o si un cortafuegos está presente, surgen nuevos problemas.

1.1.1 Interconexión de dominios

A modo de escenario de ejemplo sea una empresa que utiliza DDS para la distribución interna de contenidos. Supóngase que esta empresa ofrece ciertos servicios que requieren el acceso parcial a dichos contenidos por parte de otra compañía.

Una solución simplista al caso de uso anterior consistiría en interconectar los dos espacios de datos (el de la empresa proveedora y el de la empresa consumidora) directamente. Es decir, los administradores de la empresa proveedora podrían implementar un servicio que publique y anuncie todos los tópicos de su espacio de datos al otro. Sin embargo, el hecho de unir los dos espacios de datos sin más, generaría los siguientes problemas no desdeñables.

El primer problema estaría relacionado con la escalabilidad, ya que cada publicación disponible en un espacio de datos sería anunciada indiscriminadamente en el otro espacio de datos. Este comportamiento puede malgastar recursos en situaciones donde no haya consumidores de dicha información en el segundo espacio de datos.

Otros problemas estarían relacionados con la compatibilidad de datos. Cada espacio podría tener definido su propio (y quizás diferente) modelo de datos (*i.e.*, nombres de datos, tipos, y/o definiciones). Estos modelos podrían ser incompatibles incluso si se refieren a la misma información. Por ejemplo, la temperatura podría estar expresada en grados centígrados en un espacio de datos y en *Fahrenheit* en el otro. Por tanto, para no tener que cambiar la lógica extremo a extremo de las aplicaciones implicadas, los datos deberían ser transformados automáticamente por la infraestructura publicación-subscripción durante su transmisión entre los dos dominios. Como ventaja adicional, esto facilitaría la integración de aplicaciones DDS nuevas con otras ya existentes.

El control de acceso y la confidencialidad de la información añaden otra dimensión al problema: cierta información debería mantenerse como exclusiva de un dominio, mientras que cierta información debería ser pública.

Otro problema significativo está relacionado con hacer compatibles los diferentes requerimientos de los distintos flujos de información entre los dominios interconectados. En concreto, los administradores deberían disponer de mecanismos para establecer los requerimientos de calidad de servicio (QoS) que los dominios deben cumplir para ser interconectados.

La especificaciones de DDS actuales no abordan todos estos problemas. Consecuentemente, parte de nuestra investigación se centra en el diseño de un servicio que conecte espacios de datos DDS disjuntos (*i.e.*, un servicio de *bridging*) y que a su vez resuelva los problemas anteriormente descritos.

1.1.2 Señalización

DDS modela el intercambio de información mediante seis entidades diferentes: DomainParticipant, Tópico, Publicador, Suscriptor, DataWriter (DW), y DataReader

(DR).

DomainParticipant representa la conexión entre una aplicación y un espacio de datos compartido.

Tópico representa un flujo de información de interés, tiene asociados un nombre de tópico y un tipo de datos.

Publicador es el objeto encargado de enviar información.

Suscriptor es el objeto encargado de recibir información.

DataWriter (DW) es el objeto que utiliza la aplicación para escribir datos en un tópico. Cada DW está vinculado a un tópico concreto (y por tanto, a un tipo de datos) y pertenece a un publicador.

DataReader (DR) es el objeto que utiliza la aplicación para recibir datos de un tópico. Cada DR está vinculado a un tópico concreto (y por tanto, a un tipo de datos) y pertenece a un suscriptor.

Todas estas entidades DDS tienen QoS asociadas que especifican los aspectos no funcionales del intercambio de información.

Antes de intercambiar contenidos, las entidades DDS deben completar el *discovery*. En DDS, *discovery* es el procedimiento por el que el servicio de publicación-suscripción descubre las entidades que comparten un tópico y que además cumplen una serie de requerimientos de QoS.

Una vez que un DW descubre un DR compatible, las aplicaciones asociadas pueden comenzar a intercambiar información.

La especificación original de DDS estandarizaba un modelo de programación y una API para desarrollar aplicaciones DCPS. Pese a que DDS definía las entidades necesarias para intercambiar la información de *discovery*, ésta no especificaba el protocolo de red que las diferentes implementaciones de DDS debían usar para poder interoperar. El OMG abordó este problema con la especificación del protocolo Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol (DDS-RTPS) [69].

Para realizar el *discovery* la especificación DDS-RTPS define el denominado DDS Simple Discovery Protocol (SDP). Este protocolo fue concebido para entornos relativamente pequeños (hasta algunos miles de nodos conectados a través de una LAN). Además, un hecho relevante es que las actuales especificaciones de DDS no definen una metodología de identificación universal de entidades ni de datos.

Otro problema existente afecta a la escalabilidad. Para llevar a cabo el *discovery* la mayor parte de las implementaciones de DDS actuales se basan en usar comunicaciones *multicast* o alternativamente, usan un servidor centralizado. En este sentido, DDS-RTPS no aborda el problema del descubrimiento de entidades o contenidos en escenarios de gran escala, ni tampoco proporciona ningún mecanismo para atravesar NATs (siempre presentes en la interconexión de redes con direccionamiento público y privado). Afortunadamente, la especificación DDS-RTPS permite a las implementaciones de DDS definir sus propios protocolos de *discovery*, y por tanto es posible definir nuevos protocolos que solucionen los problemas planteados.

La introducción de un servicio de *bridging* entre dos entidades DDS (como el propuesto en esta Tesis) origina nuevos problemas. En concreto, las actuales políticas de QoS de DDS fueron diseñadas para escenarios basados en un único dominio, y quizá no tengan el comportamiento esperado en entornos en los que hay un servicio de *bridging* presente. Además, las aplicaciones requieren de nuevos mecanismos para negociar qué contenidos se propagan entre los espacios de datos interconectados.

En definitiva, la introducción de nodos intermediarios en DDS crea la necesidad de nuevos mecanismos que soporten la negociación de políticas de QoS entre nodos remotos, así como la creación de publicaciones y suscripciones a través de servicios de *bridging* para DDS. Con objeto de abordar estas cuestiones, esta Tesis define y propone el uso de un protocolo de señalización para DDS.

1.2 Objetivos

Para una mayor claridad, los objetivos de esta Tesis se han dividido en dos bloques: interconexión de espacios de datos DDS y señalización de sesión en DDS.

- **Interconexión de espacios de datos DDS:**
 - **Objetivo:** Diseñar un servicio de interconexión de espacios de datos DDS.
 - **Objetivo:** Incrementar la extensibilidad y flexibilidad de los sistemas DDS, facilitando la evolución de los sistemas basados en dicho estándar.
 - **Objetivo:** Implementar y evaluar el servicio diseñado.
 - **Objetivo:** Estudiar el impacto del servicio propuesto sobre las políticas de QoS de DDS.
- **Señalización de sesión en DDS:**
 - **Objetivo:** Diseñar un formato de identificación universal de entidades y contenidos en entornos DCPS.

- **Objetivo:** Diseñar un protocolo para el descubrimiento de entidades DCPS y para el establecimiento de sesiones en DDS.
- **Objetivo:** Evaluar la escalabilidad y robustez de la solución propuesta.

1.3 Resumen de la solución propuesta

La solución que se propone en esta Tesis consta de tres partes:

1. **Servicio de Interconexión de Dominios DDS (DDS-IS):** Proponemos una solución para la interconexión de dominios DDS que satisfaga el estándar DDS. Tras la evaluación realizada, se demuestra experimentalmente que la solución propuesta soporta la transformación de datos entre dominios y el filtrado basado en contenido, a la vez que aumenta la escalabilidad de DDS y adapta los requisitos de QoS entre las entidades comunicadas.
2. **SIP Centrado en Datos(DCSIP):** Proponemos las bases y lógica subyacente de un protocolo para la señalización de sesiones en entornos DCPS. Este protocolo resuelve el descubrimiento de contenidos y entidades DCPS en escenarios basados en DDS-IS. Además, permite la creación de sesiones entre nodos, lo que facilita la creación de rutas DCPS y el cumplimiento de QoS extremo a extremo. También proponemos un mecanismo para la identificación universal de contenidos y entidades DCPS.
3. **Uso DCPS para REsource LOcation And Discovery (RELOAD):** Proponemos una solución *Peer-to-Peer* (P2P) para el descubrimiento y acceso a contenidos mediante una aproximación DCPS. La solución propuesta se basa en el *framework* RELOAD, uno de los últimos estándares de la Internet Engineering Task Force (IETF). Nuestra solución no está limitada a escenarios basados en DDS, sino que es válida incluso si sólo algunos nodos (o incluso ningún nodo) del *overlay* soporta DDS. Se demuestra experimentalmente que nuestra solución, basada en una arquitectura P2P, proporciona una gran escalabilidad y robustez para desplegar sistemas basados en DCPS.

1.4 Estado del arte y trabajo relacionado

Esta sección estudia el estado del arte que define el contexto en el que se ha desarrollado la Tesis.

1.4.1 Modelo de interacción publicación-suscripción

La característica principal de las arquitecturas basadas en publicación-suscripción es que los consumidores de información (suscriptores) y los productores de información (publicadores) están desacoplados en tiempo, espacio y sincronización [65] [21].

Basados en el modelo de publicación-suscripción existen varios ejemplos relevantes de implementaciones de Message Oriented Middleware (MOM) [7] como Java Messaging Service (JMS) [74] y Advanced Message Queuing Protocol (AMQP) [3].

Otros ejemplos destacables son [96] y [95], trabajos en los que se propone una arquitectura publicación-suscripción para sistemas distribuidos de tiempo real basada en el *middleware* CORBA [68]. En [107] se propone una solución para el despliegue de sistemas publicación-suscripción basada en XML. Otra contribución destacable es SIENA [12], un servicio de notificación de eventos mediante publicación-suscripción basado en una arquitectura P2P. En la siguiente sección se describirán las características de éste y otros trabajos basados en P2P.

Directamente relacionado con nuestro trabajo, los autores de [75] [93] proponen DDS, un *middleware* que proporciona distribución de contenidos DCPS. Como se ha comentado previamente, DDS fue adoptado en 2004 por el OMG [73] [67] como *middleware* estándar de comunicaciones DCPS. Desde la estandarización de la especificación original en 2004, han aparecido más de ocho implementaciones diferentes del estándar [72].

DDS está en continua evolución, y en los últimos años se han adoptado varias extensiones a la especificación original. Por ejemplo, DDS-RTPS [69] define el protocolo que permite la interoperabilidad de diferentes implementaciones DDS. Otro ejemplo destacable es la especificación Extensible And Dynamic Topic Types For DDS (DDS-XTypes) [71], que permite a las aplicaciones DDS definir nuevos tipos en tiempo de ejecución.

1.4.2 Escalabilidad de los sistemas publicación-suscripción

Las tecnologías de publicación-suscripción se implementan normalmente como servicios basados en *brokers*, lo que conlleva una serie de problemas asociados a las arquitecturas centralizadas, tales como la aparición de cuellos de botella o la falta de robustez frente a errores.

Con objeto de superar estos problemas, se han propuesto muchas soluciones en la literatura. Por ejemplo, [23] propone una solución jerárquica multinivel (*clusters*

y *super-clusters*) basada en el estándar JMS para interconectar *grids*. Esta solución distribuye la carga entre los distintos *clusters*, con lo que consigue una federación de *clusters* escalable y robusta frente a fallos. Sin embargo, esta aproximación no soporta la transformación de datos, que es uno de los objetivos de nuestro trabajo. Además, esta solución no aborda el problema de la provisión de QoS, lo que dificulta su implantación en escenarios con requisitos críticos.

Los autores de [12] proponen SIENA, un servicio publicación-suscripción de notificación de eventos que utiliza dos arquitecturas para la distribución de notificaciones: cliente-servidor jerárquica y P2P. Con respecto al esquema de encaminamiento utilizado, (*i.e.*, el criterio que aplica el sistema en cada salto para tomar la decisión de transmitir los eventos desde los publicadores a los suscriptores), SIENA utiliza una aproximación basada en filtros (*i.e.*, toma la decisión de encaminamiento mediante un filtro basado en contenido situado en cada nodo). Sin embargo, este trabajo no soporta encaminamiento basado en *multicast*, que es uno de los esquemas de comunicación que soporta DDS. Además, este trabajo no considera la transformación de datos ni la provisión de QoS.

Los autores de SIENA en [13] proponen un esquema de encaminamiento que combina *broadcast* y filtrado basado en contenido para mejorar la escalabilidad del sistema. Como ocurría en el anterior trabajo, en este último tampoco se considera la transformación de los datos ni la provisión de QoS.

De forma similar, [11] propone Kyra, un esquema de encaminamiento que utiliza filtros basados en contenido y que aplica mecanismos de distribución de carga. Nuevamente, este trabajo no soporta encaminamiento basado en *multicast*, ni transformación de datos, ni provisión de QoS.

Todos los trabajos anteriormente citados utilizan filtros basados en contenido para el encaminamiento de los eventos.

A continuación se comentan otros trabajos que se basan en *multicast* para realizar el encaminamiento. Los autores de [15] proponen HOMED, un *overlay* P2P para soportar sistemas distribuidos de publicación-suscripción. HOMED organiza a los participantes de acuerdo a sus intereses con objeto de construir un árbol de distribución de eventos flexible y eficiente. Otra alternativa es [98], en la que se propone asociar suscripciones y eventos a nodos de *rendezvous*. Esta asociación se realiza en base a una combinación del identificador de dominio con el número de atributos en las suscripciones y eventos. Sin embargo, ninguna de las dos soluciones anteriores considera el problema de la transformación de datos o la provisión de QoS.

Los autores de [104] proponen particionar el espacio de eventos entre los *peers* en el sistema, y enviar los eventos y suscripciones mediante *broadcast* a todos los nodos. El coste de estos *broadcast* se reduce mediante la instalación en los nodos de filtros

asociados a las suscripciones. Como contrapartida, esta aproximación requiere en ciertos casos que todos los nodos del sistema sean contactados para instalar una suscripción. Con objeto de reducir el número de *peers* contactados, [25] propone Meghdoot, un sistema de publicación-suscripción basado en contenido basado en una infraestructura Content Addressable Network (CAN). Nuevamente ninguna de estas dos soluciones considera el problema de la transformación de datos ni la provisión de QoS.

Más recientemente, el proyecto Apache QPiD [4] propone un mecanismo de federación para AMQP [3] que incrementa la escalabilidad del sistema mediante el establecimiento de rutas dedicadas entre *brokers*. También relacionado con AMQP, [60] propone un método sistemático para configurar federaciones de *brokers* basadas en AMQP. Una aproximación diferente es propuesta en [108], donde los autores combinan *clustering*, encaminamiento basado en contenido e inundación para conseguir un sistema escalable de publicación-suscripción para MANETs. Todos los casos anteriores cubren únicamente el problema de la escalabilidad, dejando abiertas cuestiones tales como la transformación de datos.

Relacionado con DDS, los autores de [16] y [17] presentan REliable and VErsatile News delivery support for aGEncies (REVENGE). REVENGE es un *middleware* para la distribución de noticias entre clientes heterogéneos, a la vez que garantiza QoS, disponibilidad y fiabilidad. Este trabajo consta de dos contribuciones principales: un sustrato de encaminamiento P2P auto-organizable para facilitar la entrega fiable de datos entre nodos móviles, y una infraestructura de soporte DDS basada en *relays*. Esta infraestructura de soporte se caracteriza por un *overhead* reducido que permite la distribución de datos escalable en Internet. Los nodos *relay* de REVENGE permiten el intercambio de información entre distintos dominios DDS. Sin embargo, este intercambio está limitado a tópicos y tipos de datos específicos, lo que limita la flexibilidad del sistema.

Finalmente, en [77] los autores proponen una arquitectura para interconectar dominios DDS con Enterprise Service Bus (ESB), permitiendo el intercambio de información entre el mundo de los equipos empotrados y el mundo empresarial. Este trabajo se centra principalmente en incrementar la interoperabilidad de DDS mediante un sustrato de encaminamiento entre ESB y DDS. Sin embargo, no proporciona ningún mecanismo para incrementar la escalabilidad de DDS (tal como la agregación de datos propuesta en esta Tesis), ni proporciona mecanismos para la transformación de datos DDS (salvo los estrictamente necesarios para integrar ESB y DDS).

1.4.3 Descubrimiento de recursos y RELOAD

Uno de los problemas existentes de DDS es la escalabilidad de su protocolo de *discovery* en escenarios de gran escala (escenarios con miles de nodos).

En la literatura se pueden encontrar múltiples propuestas que se aprovechan de arquitecturas basadas en P2P para llevar a cabo el descubrimiento de recursos. Por ejemplo, [90] propone Spider, un *framework* para el descubrimiento de servicios Web que soporta la localización de servicios utilizando palabras clave, ontologías y patrones de comportamiento. También relacionado con el descubrimiento de servicios Web, los autores de [94] proponen un sistema de indexado escalable que incorpora mecanismos de distribución de carga para arquitecturas basadas en P2P. Sin embargo, estos dos trabajos se centran en descubrimiento de servicios Web, y por tanto no son adecuados para soportar los requerimientos específicos de los *middleware* de publicación-suscripción (por ejemplo, DDS requiere tratar a los publicadores, suscriptores y participantes de forma diferente).

SOLAR [14] propone el concepto de Descubrimiento de Recursos Sensible al Contexto (CSRD). Este trabajo, principalmente centrado en arquitecturas de tipo *context-aware*, no aborda la provisión de QoS, lo que dificulta su implantación en escenarios con requisitos críticos.

Con respecto al *discovery* de DDS, la publicación [92] propone utilizar filtros de Bloom para solucionar el problema de la escalabilidad del *discovery* en entornos DDS. Sin embargo, este trabajo no aborda otras cuestiones como el mantenimiento de un *overlay* para lograr una mayor robustez frente a fallos, o el paso a través de NATs.

En el momento de escritura de estas líneas, RELOAD [44] se encuentra en proceso de aprobación como estándar por la IETF. RELOAD es un protocolo de la IETF para el despliegue y mantenimiento de sistemas P2P en Internet.

En concreto, este protocolo facilita mecanismos para el descubrimiento de recursos y el almacenamiento y obtención de contenidos.

Con el fin de demostrar la viabilidad de RELOAD como una arquitectura P2P estable y fiable, varios trabajos han sido publicados. Por ejemplo, las prestaciones de RELOAD en entornos inalámbricos han sido evaluadas en [58], mientras que en [56] se lleva a cabo un estudio de las prestaciones de RELOAD en entornos basados en Interactive Connectivity Establishment (ICE) para el paso a través de NATs. Otra contribución es [28], que aborda el problema del descubrimiento de recursos y notificaciones en entornos Constrained Application Protocol (CoAP).

Por último, y estrechamente relacionado con nuestro trabajo, los autores de [2] proponen un protocolo de publicación-suscripción basado en RELOAD. Aunque

este trabajo proporciona las características básicas que requiere un sistema de publicación-suscripción, está totalmente basado en el *tunneling* de las muestras, lo que limita su flexibilidad y extensibilidad. Además, este trabajo no cubre el descubrimiento de entidades específicas de DDS, tales como son los *participants*.

1.4.4 DDS y SIP

Con objeto de proveer DDS de un protocolo de señalización, inicialmente se consideró la posibilidad de usar Session Initiation Protocol (SIP). SIP [83] es un protocolo especificado por la IETF [39] para la señalización de sesiones en arquitecturas cliente-servidor.

Al inicio de esta Tesis, no existían otras publicaciones que integrasen SIP y DDS. En [51] propusimos un *gateway* básico DDS-SIP para la conexión de dominios DDS. Ésta fue la primera publicación en definir el concepto de sesión DDS. En concreto, este trabajo definía una sesión DDS como un canal lógico que conecta dos dominios DDS remotos. Sin embargo, finalmente decidimos usar RELOAD –una aproximación P2P– en lugar de SIP –una aproximación cliente-servidor– porque aportaba una mayor escalabilidad al sistema.

En [26] se encuentra otro ejemplo de integración entre SIP y DDS. Este trabajo propone un sistema que integra el protocolo SIP y DDS para desplegar aplicaciones basadas en DDS sobre redes de área amplia (WANs) IP con soporte de QoS. Además, define una extensión a Session Description Protocol (sdp) que soporta la descripción de medios asociados a sesiones DDS. Como punto negativo, esta solución está limitada por la arquitectura cliente-servidor a la que SIP está ligado.

1.5 Contribuciones principales

Las principales contribuciones de esta Tesis son:

1. El diseño, implementación y evaluación de prestaciones de DDS data-space Interconnection Service (DDS-IS).
2. Un estudio del impacto de DDS-IS en la provisión de QoS.
3. La definición de identificadores universales para los dominios y tópicos DDS.
4. El diseño de un protocolo orientado a DCPS para el establecimiento de sesiones, denominado Data-Centric SIP (DCSIP).

5. El diseño, implementación y evaluación de una extensión (un *usage*) de RELOAD para proveer el descubrimiento de contenidos y entidades DCPS.

Consideramos que estas contribuciones constituyen un paso importante en la dirección de convertir DDS en una tecnología madura y sin fisuras, alineada con la visión de una Internet basada en nombres.

1.5.1 Trabajos publicados

Parte de nuestro trabajo se encuentra ya disponible para la comunidad científica a través de varias comunicaciones y publicaciones en revistas indexadas en el JCR. Adicionalmente, y como resultado del trabajo realizado, se ha solicitado una patente que se encuentra actualmente en explotación por Real-Time Innovations Inc..

Contribuciones directamente relacionadas con esta Tesis:

1. G. Pardo-Castellote, F. Sanchez, and **J. M. Lopez-Vega**, "Deploying DDS on a WAN and the GIG: The DDS router," in *Real-time and Embedded Systems Workshop. OMG*, Jul. 2009. [Online]. Disponible: <http://www.omg.org/news/meetings/GOV-WS/pr/rte.htm> [76]
2. **J. M. Lopez-Vega**, J. Povedano-Molina, and J. M. Lopez-Soler, "DDS/SIP interworking: A DDS-SIP gateway," in *Real-time and Embedded Systems Workshop*, May 2010. [Online]. Disponible: http://omg.org/news/meetings/SMCS/rt/pr/pdf/Lopez-VegaPovedano-MolinaLopez-Soler_DDS-SIP.pdf [51]
3. A. De-Campos-Ruiz, G. Pardo-Castellote, **J. M. Lopez-Vega**, and F. Crespo-Sanchez, "Patent US20110295923 - bridging data distribution services domains based on discovery data," May 2011. [Online]. Disponible: <http://www.google.com/patents/US20110295923> [18].
4. **J. M. Lopez-Vega**, J. Povedano-Molina, G. Pardo-Castellote, and J. M. Lopez-Soler, "A content-aware bridging service for publish/subscribe environments," *Journal of Systems and Software*, vol. 86, no. 1, pp. 108-124, Jan. 2013. [Online]. Disponible: <http://dx.doi.org/10.1016/j.jss.2012.07.033> [53]
5. J. Jimenez, **J. M. Lopez-Vega**, J. Maenpaa, and G. Camarillo, "A constrained application protocol (CoAP) usage for REsource LOcation and discovery (RELOAD)," *Working Draft, IETF Secretariat, Fremont, CA, USA*, Feb. 2013. [Online]. Disponible: <http://rfc-editor.org/internet-drafts/draft-jimenez-p2psip-coap-reload-03.txt> [45]

Otras contribuciones parcialmente relacionadas con 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," in *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, Jul. 2008. [Online]. Disponible: http://omg.org/news/meetings/workshops/rt_embedded_2008.htm [50]
2. J. Sanchez-Monedero, J. Povedano-Molina, **J. M. Lopez-Vega**, and J. M. Lopez-Soler, "An XML-based approach to the configuration and deployment of DDS applications," in *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, Jul. 2008. http://omg.org/news/meetings/workshops/Real-time_WS_Final_Presentations_2008/Session2/02-03_Monedero_et_al.pdf [91]
3. **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*, Feb. 2010. [52]
4. 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. [79]
5. J. Povedano-Molina, **J. M. Lopez-Vega**, and J. M. Lopez-Soler, "EMDS: an extensible multimedia distribution service," in *Real-time and Embedded Systems Workshop*, May 2010. [Online]. Disponible: http://omg.org/news/meetings/SMCS/rt/pr/pdf/Povedano-MolinaLopez-VegaLopez-Soler_EMDS.pdf [78]
6. 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*, vol. 71, no. 10, pp. 1305-1317, May 2011. [Online]. Disponible: <http://dx.doi.org/10.1016/j.jpdc.2011.05.001> [92]
7. J. M. Lopez-Soler, **J. M. Lopez-Vega**, J. Povedano-Molina, and J. J. Ramos-Munoz, "Performance evaluation of Publish/Subscribe middleware technologies for ATM (air traffic management) systems," in *Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems*, 2012. [49]

1.6 Estructura del documento

A continuación se describe como se ha estructurado el presente documento.

En el Capítulo 2 se estudian las tecnologías que supusieron el punto de partida de esta Tesis.

La solución propuesta ha sido dividida en tres capítulos que se corresponden a la división descrita en la Sección 1.3. Cada uno de dichos capítulos incluye su propia introducción, casos de uso, diseño, experimentos, resultados, discusión de los resultados y conclusiones. Esta estructura facilita la lectura del documento, ya que cada capítulo está autocontenido y puede ser leído de forma independiente.

En concreto, el Capítulo 3 incluye el diseño, implementación y evaluación de prestaciones de un servicio de interconexión de dominios DDS; asimismo incluye un estudio del impacto del servicio en la provisión de QoS.

En el Capítulo 4 se presenta el diseño de un protocolo orientado a DCPS para el establecimiento de sesiones, llamado DCSIP.

En el Capítulo 5 se presenta el diseño, implementación, y evaluación de una nueva extensión (*usage*) para RELOAD que proporciona mecanismos para el descubrimiento de contenidos y entidades DCPS.

Por último, en el Capítulo 6 se discuten trabajos relacionados, y en el Capítulo 7 se resumen las principales conclusiones de nuestro trabajo.

Introduction

CURRENT uses of the Internet are mostly built on request-response exchanges between pairs of computers. This model has been very successful at providing shared remote access to resources and point-to-point communications for applications including email, remote file and database access, web access, etc.

Transmission Control Protocol (TCP), the most widely used Internet protocol, is also geared towards providing reliable point-to-point communications, to the extent that each TCP message embeds the location of the sending and receiving endpoints.

The advent of ubiquitous Internet connectivity and cheap network-connected devices has greatly expanded the communication needs of applications using the Internet. For these new kinds of applications the request-response communications model is cumbersome and limited whereas other communication models, such as publish-subscribe [21], provide a more natural fit. The nature and historical context of this problem is well described in Van Jacobson *et al.* seminal work [41]:

The engineering principles and architecture of today's Internet were created in the 1960s and 70s. The problem networking aimed to solve was resource sharing [...]. The communication model that resulted is a conversation between exactly two machines, one wishing to use the resource and one providing access to it. Thus IP packets contain two identifiers (addresses), one for the source and one for the destination host, and almost all the traffic on the Internet consists of (TCP) conversations between pairs of hosts.

In the 50 years since the creation of packet networking, computers and their attachments have become cheap, ubiquitous commodities. The connectivity offered by the Internet and low storage costs enable access to a staggering amount of new content – 500

exabytes created in 2008 alone [24]. People value the Internet for **what** content it contains, but communication is still in terms of **where**.

We see a number of issues that affect users arising from this incompatibility between models.

Availability: Fast, reliable content access requires awkward, pre-planned, application-specific mechanisms like CDNs and P2P networks, and/or imposes excessive bandwidth costs.

Security: Trust in content is easily misplaced, relying on untrustworthy location and connection information.

Location-dependence: Mapping content to host locations complicates configuration as well as implementation of network services. The direct, unified way to solve these problems is to replace where with what. Host-to-host conversations are a networking abstraction chosen to fit the problems of the 60s. We argue that named data is a better abstraction for today's communication problems than named hosts.

In contrast to the request-response interaction model, in which a client sends a request and a server returns a response, this Thesis –that in a sense is aligned with Jacobson's vision– focuses on the publish-subscribe interaction model. In the publish-subscribe model information producers (publishers) are decoupled from information consumers (subscribers) via the publish-subscribe service. Publishers generate events or update a shared information space, subscribers express their interest in certain patterns of events or parts of the information space, and the publish-subscribe service delivers the desired information from publishers to subscribers.

This vision is well aligned with recent projects aimed at defining a future publish-subscribe-based Internet that substitutes the current Internet design, in particular Publish-Subscribe Internet Routing Paradigm (PSIRP) [20] [22] and Publish Subscribe Internet Technology (PURSUIT) [82].

1.1 Motivation

In 2004, the Object Management Group (OMG) [73] adopted Data Distribution Service (DDS) [67], a specification that standardizes data-centric publish-subscribe communications. This specification defines a standardized Application Programming Interface (API) for exchanging information following a Data-Centric Publish-Subscribe (DCPS) model. In this model, publishers and subscribers exchange information through a distributed cache known as data-space (also known as DDS domain). Another relevant characteristic of DCPS is that exchanged information

is not opaque to the middleware. In this way, DDS is able to provide advanced functionalities such as data-caching or content-based-filtering.

The DDS standard is still growing. Proof of the activity around this standard is the fact that just in the last two years the OMG has published four of the six documents currently constituting the DDS standard family. As a consequence of the relative novelty of DDS, there are still several problems to solve.

DDS was originally designed for isolated Local Area Networks (LANs) in which data sources (publishers) and data consumers (subscribers) are located in the same geographical location.

Problems arise if a subscriber has interest in data being originated in a different data-space, especially if that data-space is separated by a bandwidth-constrained network, or if a Network Address Translation (NAT) or firewall service is present.

1.1.1 Domain interconnection

Consider an example scenario with a company that relies on DDS for delivering some data-content internally. In addition, this company provides certain services that require to expose a subset of that data-content to another company.

A simplistic solution to the previous use case would be to merge the two data-spaces. That is, to implement a service that publishes and announces all the topics from one data-space to the other. However, the direct merging of the data-spaces would create some significant problems.

The first problem is related to scalability. Every publication available in one data-space will be indiscriminately announced in the other data-space. This may be wasteful in resources in situations where there are no consumers to that information on the second data-space.

Other problems are concerned to data compatibility. For instance, each data-space may have its own (possibly different) data model (*i.e.*, data names, types, and/or definitions). These models could be incompatible even if they refer to the same data items. For example, temperature data might be expressed in Celsius in one data-space and in Fahrenheit in the other one. Therefore, in order to keep the end-to-end application logic unchanged, data should be automatically transformed by the publish-subscribe infrastructure when bridging data-spaces with different data models. As an additional benefit, this will ease the integration between new and legacy DDS applications.

Access control and information confidentiality adds another dimension to this problem: Some parts of the data should be confined in a data-space whereas other

data should be public.

Another relevant issue is related to the possibility of having different delivery requirements for the information that flows between and within the different interconnected data-spaces. For example, system administrators should be able to establish the Quality of Service (QoS) requirements that domains must meet in order to be bridged.

The current DDS specifications do not address any of these domain interconnection issues. Consequently, one aspect of this research focused on the design of a service that connects disjoint DDS data-spaces (*i.e.*, a bridging service) while also solving the problems stated above.

1.1.2 Signaling

DDS models information exchange using six different entities: DomainParticipant, Topic, Publisher, Subscriber, DataWriter (DW), and DataReader (DR).

DomainParticipant represents the connection from an application to a shared data-space.

Topic represents a stream of information of interest, it has an associated topic name and data-type.

Publisher is the object responsible for sending information.

Subscriber is the object responsible for receiving information.

DataWriter (DW) is the object the application uses to write data to a specific topic. Each DW is bound to a specific topic (and therefore data-type) and belongs to a publisher.

DataReader (DR) is the object the application uses to receive data on a specific topic. Each DR is bound to a topic (and therefore data-type) and belongs to a subscriber.

All these DDS entities have QoS attached to them specifying non-functional aspects of the information exchange.

In order to exchange data-content, DDS entities must discover each other. DDS includes a discovery mechanism. DDS discovery is the process used by the publish-subscribe service to find the entities which share a particular topic and meet a set of QoS requirements.

Once a DW discovers a matching DR the corresponding applications can start exchanging data.

Originally, the DDS specification standardized the programming model and API for developing DCPS applications. Despite of the fact that it defined the entities for exchanging discovery information, it did not specify the network protocol used by DDS implementations in a way that would allow different implementations to interoperate. The OMG addressed this issue by specifying the Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol (DDS-RTPS) [69].

The DDS-RTPS specification defines the so called DDS Simple Discovery Protocol (SDP) to perform the discovery. This discovery protocol was designed for relatively small environments (up to a few thousands nodes connected through a LAN). In this regard, the current DDS specifications do not address the problem of universal entity identification and universal data-content identification.

Another existing limitation impacts scalability. Most of current DDS implementations rely on using multicast or a centralized server for discovery. In this sense, DDS-RTPS does not address the problem of entity or data-content discovery in large-scale deployments, nor provides any mechanism for performing NAT traversal. Fortunately, the DDS-RTPS specification allows DDS implementations to define their own discovery protocols, therefore is possible to define new protocols that address these issues.

The introduction of a bridging service between DDS entities (as the one proposed in this Thesis) creates new problems. In particular, current DDS QoS policies were designed for single-domain-based scenarios, and they may behave unexpectedly in environments where a bridging service is present. In addition, applications need new mechanisms for negotiating what contents are propagated through different data-spaces. In short, the introduction of intermediate nodes in DDS creates the need of new mechanisms that support the negotiation of QoS policies among remote nodes, along with the creation of publications and subscriptions across DDS bridging services. In order to address these issues, this Thesis defines and proposes the use of a signaling protocol for DDS.

1.2 Goals

The main objectives of the Thesis are organized in two categories: DDS data-space interconnection and session signaling.

- **DDS data-space interconnection:**
 - **Objective:** To design a data-space bridging service for DDS.

- **Objective:** To increase the extensibility and flexibility of DDS systems, easing DDS systems evolution.
 - **Objective:** To implement and evaluate the designed data-space bridging service for DDS.
 - **Objective:** To study the impact of the proposed service in DDS QoS.
- **Session signaling in DDS:**
 - **Objective:** To design an universal entity and data-content naming format for DCPS environments.
 - **Objective:** To design a protocol for DCPS entities discovery and session establishment in DDS.
 - **Objective:** To evaluate the scalability and robustness of the proposed solution.

1.3 Solution overview

The solution proposed in this Thesis consists of three parts:

1. **DDS data-space Interconnection Service (DDS-IS):** We propose a solution for the interconnection of DDS data-spaces that is compliant with DDS specification. We experimentally demonstrate that it provides data transformation between data-spaces, performs content-based filtering, increases DDS scalability, and provides QoS adaptation between remote entities.
2. **Data-Centric SIP (DCSIP):** We propose the fundamentals and rationale of a protocol for session signaling in DCPS environments. This protocol solves DCPS entity and data-content discovery in DDS-IS-based deployments. It also allows the creation of sessions between nodes, which eases the creation of DCPS routes and the enforcement of QoS at end-to-end level. We also propose a mechanism for universal identification of DCPS entities and data-content.
3. **DCPS usage for Resource Location And Discovery (RELOAD):** We propose a Peer-to-Peer (P2P) solution for discovering and accessing to content using a DCPS approach. Our solution is based on the RELOAD framework, one of the latest standards of the Internet Engineering Task Force (IETF). Our solution is not limited to DDS-based scenarios, and it is valid even if only a few (or none) of the nodes in the overlay supports DDS. We experimentally demonstrate that our solution, based on a P2P architecture, features great scalability and robustness for deploying DCPS-based systems.

1.4 State of the art and related work

This section studies the state of the art that provides the context for the research carried out in this Thesis.

1.4.1 Publish-subscribe interaction model

The key feature of publish-subscribe-based architectures is that information consumers (subscribers) and information producers (publishers) are decoupled in time, space, and synchronization [65] [21].

Several Message Oriented Middleware (MOM) [7] implementations, which are based on asynchronous message-passing, use the publish-subscribe model for exchanging data. For example, Java Messaging Service (JMS) [74] and Advanced Message Queuing Protocol (AMQP) [3] are MOM-based technologies that allow applications to deliver messages using a publish-subscribe approach.

Other relevant approaches are also remarkable. For instance, [96] and [95] propose a publisher-subscriber architecture for distributed real-time systems based on CORBA middleware [68], and [107] proposes an XML-based solution for deploying publish-subscribe systems. Another interesting solution is SIENA [12], a publish-subscribe event notification service that takes advantage of a P2P architecture. We will study this and other P2P-based approaches in the next section.

Directly related to our work, the authors of [75] [93] propose DDS, a middleware that provides data-centric publish-subscribe content distribution.

DDS was adopted in 2004 by the OMG [73] [67] as the standard middleware for DCPS communications. Since the original DDS specification in 2004, more than eight different implementations of the standard have appeared [72].

The DDS standard is still growing, and several extensions to the original specification have been adopted over the last years. For example, DDS-RTPS [69] defines a protocol that allows different DDS implementations to interoperate. Another noteworthy example is the Extensible And Dynamic Topic Types For DDS (DDS-XTypes) [71] specification, which allows DDS applications to define new types in runtime. However, and due to the novelty of the standard, several problems, such as the interconnection of disjoint domains, remain unsolved.

1.4.2 Scalability in publish-subscribe systems

Publish-subscribe technologies are usually implemented as brokered services. This leads to well known issues that derive from centralized architectures such as single point of failure and bottlenecks.

To cope with these issues, many solutions have been proposed in the literature. For instance, [23] proposes a hierarchical multi level (clusters and super-clusters) JMS compliant approach for interconnecting grids. It features scalability, load-balancing, and fault-tolerant federation. However, this approach does not support data transformation, which is one of our initial objectives. Moreover, it does not take into account QoS related issues, making difficult its usage in critical scenarios.

The authors of [12] propose SIENA, a publish-subscribe event notification service that utilizes two different architectures for distributing event notifications: hierarchical client-server and P2P. Regarding the routing scheme (*i.e.*, the criteria that the system follows in each hop to decide how forward samples from publishers to subscribers), SIENA uses a filter-based approach (*i.e.*, it makes routing decisions via successive content-based filtering at all nodes from source to destination). However, this work does not support multicast-based routing, which is one of the communication schemes that DDS supports. Moreover, this work does not consider data transformation nor QoS enforcement.

From the authors of SIENA, [13] proposes a routing schema that combines broadcast and content-based filtering for improving the scalability of the system. As in the previous work, it does not consider data transformation nor QoS enforcement.

Similarly, [11] proposes Kyra, a routing scheme based on content-based filtering that applies load balancing mechanisms. Again, this work does not support multicast-based routing, data transformation, nor QoS enforcement.

All the above-mentioned work utilize a content-based filtering approach.

There are also existing approaches that use a multicast-based routing scheme. For example, the authors of [15] propose HOMED, a P2P overlay for supporting distributed publish-subscribe systems. It constructs a flexible and efficient event dissemination tree by organizing participants based on their interest. Another interesting work is [98], which proposes an alternative for creating the multicast dissemination tree. In particular, it proposes to map subscriptions and events to rendezvous nodes; for that, it uses a combination of the domain schema identifier and the numbers of attributes in the subscriptions or the events. None of the above two solutions considers the problem of data transformation nor QoS enforcement.

The authors of [104] propose to partition the event space among the peers in the system, and to broadcast events and subscriptions to all nodes in the system.

These broadcasts are attenuated by filters installed at peer nodes depending on the subscriptions stored in the system. However, this approach may require all peers in the system to be contacted to install a subscription. To reduce the number of peers to be contacted, [25] proposes Meghdoot, a content-based publish-subscribe system built on top of Content Addressable Network (CAN) infrastructure. Again, none of these two solutions considers the problem of data transformation nor QoS enforcement.

More recently, the Apache QPiD project [4] includes a federation mechanism for AMQP [3] that increases the scalability of the system by establishing dedicated routes between brokers. Also related to AMQP, [60] provides a systematic method for configuring broker federation in AMQP-based systems. A different approach is included in [108], where the authors combine clustering, content-based routing, and document flooding for providing scalable publish-subscribe communications for MANETs. In all these cases, they deal only with scalability problem, leaving open issues such as data transformation.

Related to DDS, the authors of [16] and [17] present REliable and VErsatile News delivery support for aGencies (REVENGE). REVENGE is a middleware used to distribute breaking news among heterogeneous clients while guaranteeing QoS, availability, and reliability. This work has two main contributions: a self-organizing P2P routing substrate to facilitate reliable data delivery between mobile nodes, and a relay-based DDS support infrastructure with limited overhead to enable scalable, Internet-wide data dissemination. REVENGE relay nodes allow for the exchange of information among different DDS domains. However, this communication is bound to a specific topic/data-type, which limits the flexibility of the system.

Finally, in [77] the authors propose an architecture to interconnect DDS data-spaces with Enterprise Service Bus (ESB), thus allowing to exchange information between the embedded world and the enterprise system. This work is mainly focused on increasing the interoperability of DDS by providing a routing substrate between ESB and DDS. However, it does not provide any mechanism to increase the scalability on the DDS system (like the data aggregation proposed in this Thesis), and does not provide mechanisms for performing DDS data transformations (except the strictly necessary for integrating ESB and DDS).

1.4.3 Resource discovery and RELOAD

One of the current issues of DDS is the scalability of its discovery protocol in large-scale deployments (deployments with thousands of nodes).

In the literature we can find several proposals that take advantage of P2P-based architectures for performing resource discovery. For example, [90] proposes Spi-

der, a framework for Web Service discovery. It supports the lookup of Web Services based on keywords, ontologies, or behavior. Also related to Web Service discovery, the authors of [94] propose a scalable indexing system for P2P-based architectures that incorporates load-balancing mechanisms. However, these two works are focused in Web Service discovery, and therefore they are not suited to support the specific requirements of a publish-subscribe middleware (for example, DDS requires to treat publishers, subscribers, and participants differently).

SOLAR [14] proposes the Context-Sensitive Resource Discovery (CSRD). This work, mainly focused on context-aware architectures, does not take into account QoS related issues, making it difficult to be used in more critical scenarios.

Regarding discovery in DDS, the paper [92] addresses the problem of scalable discovery in DDS environments by applying Bloom filters. However, this work only focuses on the discovery problem itself, not addressing other issues such as maintaining an overlay to provide fault-tolerance, or NAT-traversal.

In the time of writing, the IETF is in the process of approving RELOAD [44] as standard. RELOAD is an IETF protocol for building and maintaining P2P systems on the Internet. In particular, it provides mechanisms for resource discovery and for data-content storage and retrieval.

A number of papers have aimed to demonstrate the feasibility of RELOAD as an stable and reliable P2P architecture. For instance, RELOAD performance in wireless environments has been studied in [58]. In [56], a study of the performance of RELOAD on an Interactive Connectivity Establishment (ICE)-based environment for NAT traversal is done. Another contribution is [28], which addresses the problem of resource discovery and notification in Constrained Application Protocol (CoAP) environments.

Finally, and very close to our work, the authors of [2] propose a publish-subscribe protocol built atop of RELOAD. Although it provides the basic features needed for a publish-subscribe system, it relies completely on tunneling publish-subscribe samples, which hinders its flexibility and extensibility. Moreover, this work does not cover the discovery of specific DDS entities, such as participants.

1.4.4 DDS and SIP

In order to provide DDS of a signaling protocol, we initially consider using Session Initiation Protocol (SIP). SIP [83] is an IETF [39] protocol specification for session signaling in client-server-based architectures.

At the beginning of this Thesis, there were no other publications that integrated SIP and DDS. In [51] we proposed a basic DDS-SIP gateway to connect remote DDS

domains. This was the first publication that defined the concept of DDS session. This work defined DDS session as a logical channel that connects two remote DDS domains. However, we finally decided to use RELOAD –a P2P approach– instead of SIP –a client-server approach– because it provides greater scalability.

We can find another example of SIP-DDS integration in [26]. This paper proposes a system that integrates SIP and DDS for supporting DDS-based applications over QoS-enabled IP WANs. It also defines an extension to Session Description Protocol (sdp) that supports the media description of DDS sessions. As a downside, this solution is limited by the client-server architecture SIP is bound to.

1.5 Main contributions

The main contributions of the Thesis are:

1. The design, implementation, and performance evaluation of a DDS data-space Interconnection Service (DDS-IS).
2. A study of the impact of DDS-IS in QoS enforcement.
3. The definition of universal identifiers for DDS domains and topics.
4. The design of a DCPS-oriented protocol for session establishment, Data-Centric SIP (DCSIP).
5. The design, implementation, and evaluation of a new RELOAD usage for providing DCPS entity and data-content discovery.

We definitively believe these contributions constitute an important step toward making DDS a mature and seamless technology, aligned with the envisaged named-based Internet.

1.5.1 Published works

Part of our work is already available to the research community through several publications in indexed journals and conference proceedings. In addition, as a result of the conducted work we have applied for a patent that is currently being pursued by Real-Time Innovations Inc.

Contributions directly related to this Thesis:

1. G. Pardo-Castellote, F. Sanchez, and **J. M. Lopez-Vega**, "Deploying DDS on a WAN and the GIG: The DDS router," in *Real-time and Embedded Systems Workshop*. OMG, Jul. 2009. [Online]. Available: <http://www.omg.org/news/meetings/GOV-WS/pr/rte.htm> [76]
2. **J. M. Lopez-Vega**, J. Povedano-Molina, and J. M. Lopez-Soler, "DDS/SIP interworking: A DDS-SIP gateway," in *Real-time and Embedded Systems Workshop*, May 2010. [Online]. Available: http://omg.org/news/meetings/SMCS/rt/pr/pdf/Lopez-VegaPovedano-MolinaLopez-Soler_DDS-SIP.pdf [51]
3. A. De-Campos-Ruiz, G. Pardo-Castellote, **J. M. Lopez-Vega**, and F. Crespo-Sanchez, "Patent US20110295923 - bridging data distribution services domains based on discovery data," May 2011. [Online]. Available: <http://www.google.com/patents/US20110295923> [18].
4. **J. M. Lopez-Vega**, J. Povedano-Molina, G. Pardo-Castellote, and J. M. Lopez-Soler, "A content-aware bridging service for publish/subscribe environments," *Journal of Systems and Software*, vol. 86, no. 1, pp. 108-124, Jan. 2013. [Online]. Available: <http://dx.doi.org/10.1016/j.jss.2012.07.033> [53]
5. J. Jimenez, **J. M. Lopez-Vega**, J. Maenpaa, and G. Camarillo, "A constrained application protocol (CoAP) usage for REsource LOcation and discovery (RELOAD)," *Working Draft, IETF Secretariat, Fremont, CA, USA*, Feb. 2013. [Online]. Available: <http://rfc-editor.org/internet-drafts/draft-jimenez-p2psip-coap-reload-03.txt> [45]

Other contributions partially related to 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*, Jul. 2008. [Online]. Available: http://omg.org/news/meetings/workshops/rt_embedded_2008.htm [50]
2. J. Sanchez-Monedero, J. Povedano-Molina, **J. M. Lopez-Vega**, and J. M. Lopez-Soler, "An XML-based approach to the configuration and deployment of DDS applications," in *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, Jul. 2008. http://omg.org/news/meetings/workshops/Real-time_WS_Final_Presentations_2008/Session2/02-03_Monedero_et_al.pdf [91]
3. **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*, Feb. 2010. [52]

4. 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. [79]
5. J. Povedano-Molina, **J. M. Lopez-Vega**, and J. M. Lopez-Soler, "EMDS: an extensible multimedia distribution service," in *Real-time and Embedded Systems Workshop*, May 2010. [Online]. Available: http://omg.org/news/meetings/SMCS/rt/pr/pdf/Povedano-MolinaLopez-VegaLopez-Soler_EMDS.pdf [78]
6. 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*, vol. 71, no. 10, pp. 1305-1317, May 2011. [Online]. Available: <http://dx.doi.org/10.1016/j.jpdc.2011.05.001> [92]
7. J. M. Lopez-Soler, **J. M. Lopez-Vega**, J. Povedano-Molina, and J. J. Ramos-Munoz, "Performance evaluation of Publish/Subscribe middleware technologies for ATM (air traffic management) systems," in *Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems*, 2012. [49]

1.6 Document structure

The remainder of the document is organized as follows.

In Chapter 2 we study the technologies that provided the starting point of this Thesis.

We have divided the proposed solution in three chapters that correspond to the division described in Section 1.3. Each one of these chapters includes its own introduction, motivating use cases, design, experiments, results, discussion of results, and conclusions. This structure eases the reading of the document, as each chapter is self-contained.

In particular, Chapter 3 includes the design, implementation, and performance evaluation of a DDS data-space Interconnection Service; it also includes an study of the impact of this service in QoS enforcement.

In Chapter 4 we present the design of a DCPS-oriented protocol for session establishment, named DCSIP.

In Chapter 5 we present the design, implementation, and evaluation of a new usage for RELOAD that provides DCPS entity and data-content discovery.

Finally, in Chapter 6 we discuss related work, and in Chapter 7 we summarize the main conclusions of our work.

Chapter 2

Background

IN this chapter we study the technologies that provided the starting point of this Thesis.

First, we study the publish-subscribe interaction model, which is an alternative to the request-response model. In particular, we study the DDS specification, which is a standardized publish-subscribe middleware for data dissemination on real-time distributed systems.

Regarding signaling and resource discovery, we study SIP, an application protocol for session signaling, and RELOAD, a standardized P2P-based framework for highly scalable resource discovery.

Finally, we introduce CoAP, a HTTP-inspired protocol for exchange data-content between highly constrained devices.

2.1 Publish-subscribe interaction model

Over the last few years, publish-subscribe communication systems are gaining momentum and popularity, as verified beyond doubt by the existence of successful projects like PSIRP [20] [22], and PURSUIT [82], which is a continuation of PSIRP. These projects are aimed to define a future publish-subscribe-based Internet that substitutes the current Internet design, based on TCP/IP [101].

In contrast to the request-response interaction model, in which a client sends a request and a server returns a response, in the publish-subscribe interaction model



Figure 2.1: Request-response and publish-subscribe interaction models.

information producers (publishers) are decoupled from information consumers (subscribers) via the publish-subscribe service. Publishers generate events or update a shared information space, subscribers express their interest in certain patterns of events or parts of the information space, and the publish-subscribe service delivers the desired information from publishers to subscribers. (see Figure 2.1 [48]).

The key feature of publish-subscribe-based architectures is that information consumers (subscribers) and information producers (publishers) are triply decoupled [21].

In particular, publish-subscribe applications are decoupled in space (publishers and subscribers can be located anywhere), in time (there is no need of simultaneous end-point availability), and in synchronization/flow (publishers are not blocked while producing events, and subscribers can get notified about the occurrence of some event while performing some concurrent activity).

We can classify publish-subscribe systems according to the mechanism that they use for specifying what are the events of interest [21] (*i.e.*, the scheme). The most widely used schemes are topic-based, content-based, and type-based. In topic-based schemes, events are classified according to a label, the topic. In content-based schemes, events are classified using the content of the events. Finally, in type-based schemes, events are distinguished according to their structure.

Several MOM [7] implementations, which are based on asynchronous message-passing, use the publish-subscribe model for exchanging data. For example, JMS [74] and AMQP [3] are MOM-based technologies that allow applications to deliver messages using a publish-subscribe approach.

Other relevant approaches are also remarkable. For instance, [96] and [95] propose a publisher-subscriber architecture for distributed real-time systems based on CORBA middleware [68], and [107] proposes an XML-based solution for deploying publish-subscribe systems. Another relevant work is SIENA [12], a publish-subscribe event notification service which was studied in Section 1.4.2.

Directly related to our work, the authors of [75] [93] propose DDS, a middleware that provides data-centric publish-subscribe content distribution. In the next sec-

tion we study OMG DDS, which is the standard specification for this middleware.

2.2 Data Distribution Service (DDS)

DDS [67] is a specification adopted by the OMG [73]. DDS standardizes DCPS communications in distributed scenarios. For more information about the DCPS model, please refer to Section 2.2.1.

Since the original DDS specification in 2004, several implementations of the standard have appeared [72]: RTI Connexant [89], a COTS implementation of DDS which supports multiple languages and platforms; OpenSplice [81], a partially open implementation of DDS which also supports multiple languages and platforms; OpenDDS [62], an open source C++ implementation; MilSOFT DDS [61], a COTS implementation of DDS; OCERA ORTE [63], an open source implementation of RTPS 1.0.; CoreDX [105], a COTS implementation of the DDS specification; InterCOM DDS [47], an implementation of the DDS minimum profile; MicroDDS [30], a DDS implementation for small devices, micro-controllers and embedded systems; and BEE-DDS [100], a Java implementation of the OMG DDS standard.

Over the last few years, DDS has been deployed in many contexts ranging from mission critical systems to financial environments [85] [86] [19] [87] [80]. All these scenarios have in common that different collocated entities exchange data samples coming from different sources (for example, position sensors, stocks, etc.) in a well controlled environment (*i.e.*, a data-space). In addition, this information is exchanged with certain guarantees such as reliability or deterministic time-delivery.

DDS uses a data-centric publish-subscribe model along with a rich set of QoS profiles that can be tuned to fit the communication demands of each specific scenario.

DDS specification defines two different interface levels [75] and an interoperability layer [69]:

- The **Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol (DDS-RTPS)** interoperability layer. This layer is based on the Real Time Publish Subscribe (RTPS) protocol, and allows different DDS implementations to interoperate. This layer defines the discovery protocol, data representation format, and message format DDS applications must use in order to interoperate.
- The mandatory **Data-Centric Publish-Subscribe (DCPS)** interface level. This defines a programming model and API for developing applications using a data-centric publish-subscribe approach.

- The optional **Data Local Reconstruction Layer (DLRL)** interface level. This layer acts as an interface between the application layer and the DCPS layer. DLRL is object oriented and was specified to ease the integration of DCPS in user applications. This level allows developers to access to DDS data-content through object attributes.

In what follows under this section, we focus on studying the DCPS layer and the DDS-RTPS protocol, which have a key role in our research.

2.2.1 Data Centric Publish Subscribe layer (DCPS)

The DCPS layer specification [67] defines the Data-Centric Publish-Subscribe (DCPS) programming model and its associated API for developing distributed applications. DCPS programming model is based in two main concepts: data-centricity and publish-subscribe.

Data-centric oriented middleware, such as DDS, contrasts with MOM in that message content is not opaque to the middleware. Using the functionality provided by the DCPS layer, DDS is able to interpret exchanged data-samples. In this way, DDS is able to provide advanced functionalities such as data-caching or content-based-filtering.

The DCPS programming and communication model is based on the publish-subscribe paradigm, and therefore DCPS-based applications are decoupled in space, time, and synchronization.

In addition, DCPS-based applications are also decoupled in platform: DCPS developers can deploy distributed systems using diverse operating systems, hardware architectures, and languages.

The DCPS layer defines the following concepts:

Data-space: Also known as domain, it is an aggregation of data of different data-types and sources. More precisely, a data-space is a distributed cache, whose content is accessed and updated by subscribers and publishers.

Topic: A portion of a data-space that groups data of the same data-type. Although we could think that DDS uses a topic-based scheme, this is not strictly true, since topics are usually associated to a type, which is typical of type-based schemes. Moreover, DDS supports the use of content-based filtering, which is a typical characteristic of content-based schemes.

Publisher: An entity that pushes data-content updates to one or more topics.

Subscriber: An entity that is interested in data-content updates from one or

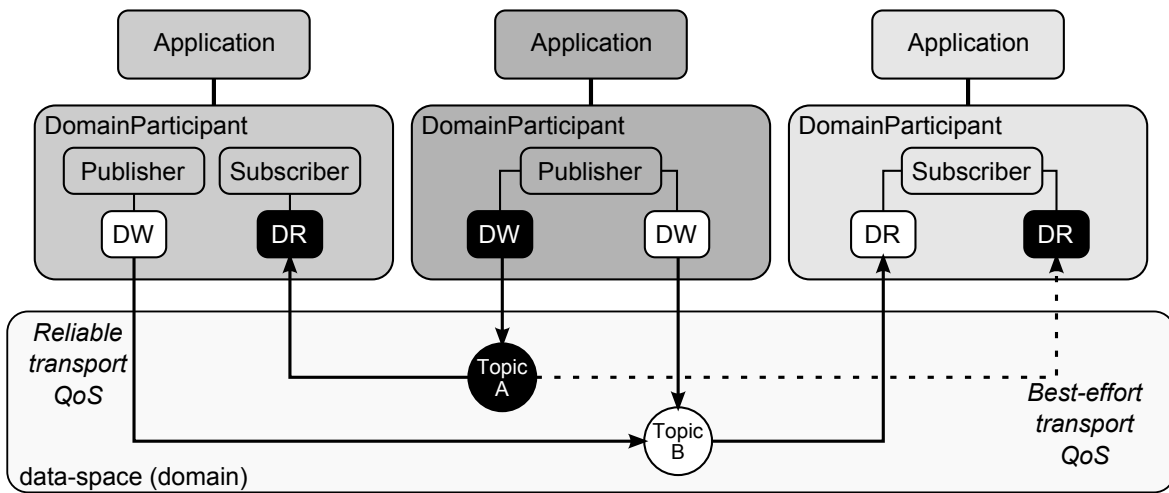


Figure 2.2: DDS concepts and entities.

more topics.

According to DCPS, publishers and subscribers exchange data-content within a particular shared data-space. DDS applications access and update the data-content in this data-space using topic-specific endpoint entities, respectively called **DRs** and **DWs**. Applications access to all these entities through **DomainParticipants**, which are the interface between applications and DDS data-spaces.

Under the DCPS model, when a publisher wants to update data-content belonging to certain topic, the middleware assumes responsibility for managing its delivery to the proper interested peers (*i.e.*, to the associated subscribers). As a result, since DDS relieves the application from burden of transmitting and managing the data, the application logic is drastically simplified. Figure 2.2 illustrates main DDS concepts and relationships.

DDS states that within a data-space data objects are completely identified by a topic name, a data-type, and optionally by a key. This key is useful to identify different instances of the same topic.

In order to exchange data-content, DWs and DRs (referred to as endpoints) have to find compatible endpoints. The process of finding compatible entities is known as discovery. DDS performs discovery using a set of built-in publications (*i.e.*, a set of mandatory-to-implement publications), which are handled using the so-called Built-in DataReaders (B-DRs) and Built-in DataWriters (B-DWs) (see Figure 2.3).

The discovery function allows DDS applications to automatically discover available (and compatible) publications and subscriptions.

Another feature of DCPS is the support of QoS policies. A QoS policy repre-

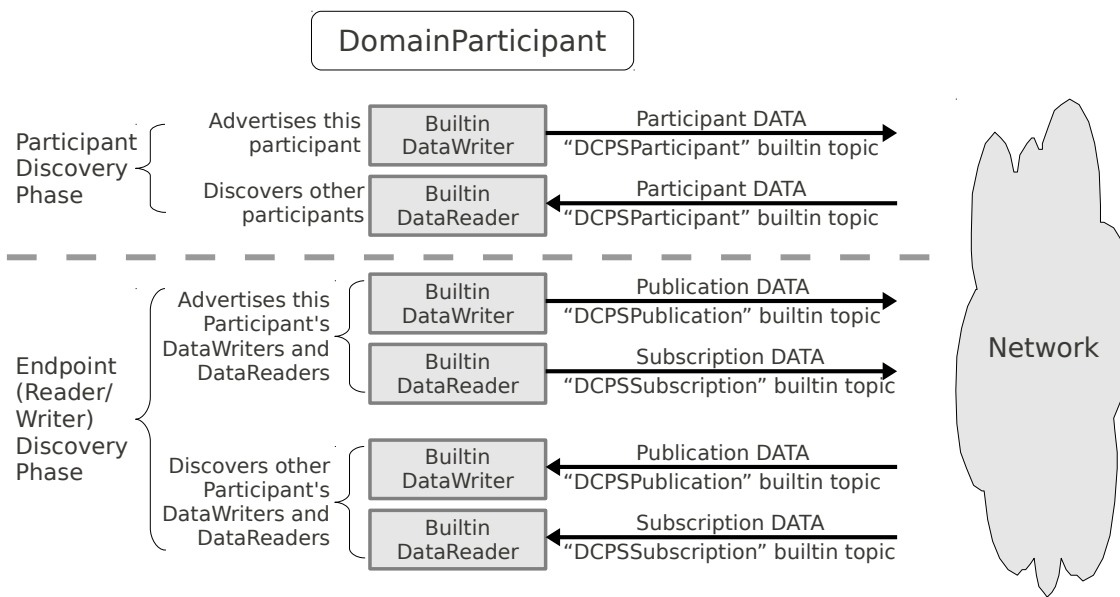


Figure 2.3: DCPS built-in entities for discovery purposes.

sents an agreement among a group of DDS entities. This agreement determines the behavior of the middleware, and releases the application logic of certain tasks, such as communication reliability. In case of QoS infringement, DDS detects it and optionally triggers proper actions. Each DDS entity has its own QoS policies and data-caching attributes. As an example, a DW and a DR can negotiate the level of reliability of the communications (best effort or reliable delivery) by using the RELIABILITY QoS. It is also possible to guarantee a maximum period between data updates (DEADLINE QoS), or to filter samples according to their content or timing.

2.2.2 Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol (DDS-RTPS)

The original DDS specification did not enforce any particular lower level implementation requirement or recommendation, such as the use of a particular underlying transport protocol. For the sake of vendors interoperability, the OMG defined the so-called Real-Time Publish-Subscribe Wire Protocol Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol (DDS-RTPS) [69].

DDS-RTPS is based on Real-Time Publish-Subscribe (RTPS), a protocol originally approved by the International Electrotechnical Commission (IEC) [32] as part of the Real-Time Industrial Ethernet Suite IEC-PAS-62030 [31]. After applying some modifications, the OMG adopted RTPS as the standard wire protocol for DDS. DDS-RTPS specifies the data representation, message format, and discovery proto-

col that DDS implementations must use in order to interoperate.

In order to exchange data-content, DDS entities must discover each other. DDS includes a discovery mechanism. DDS discovery is the process used by the publish-subscribe service to find the entities which share a particular topic and meet a set of QoS requirements. One of the goals of DDS-RTPS is to allow entities belonging to different DDS implementations to perform discovery. DDS-RTPS discovery re-utilizes most of the existing DDS core entities. Specifically, the correspondence between DDS-RTPS and DDS entities is shown in Table 2.1.

Table 2.1: Correspondence DDS and DDS-RTPS entities.

DDS entity	DDS-RTPS entity
DomainParticipant	Participant
DataWriter	Writer
DataReader	Reader
Built-in DataWriter	ENTITYID_SPDP_BUILTIN_*_WRITER
Built-in DataReader	ENTITYID_SPDP_BUILTIN_*_READER

According to DDS-RTPS, any discovery protocol must include the following phases: Participant Discovery Protocol (PDP) and Endpoint Discovery Protocol (EDP). The purpose of PDP is to discover new participants in the data-space. Whenever a new participant is discovered, the EDP procedure is triggered to exchange local and remote endpoints information between two participants.

Different implementations may choose to support multiple PDPs and EDPs, possibly vendor-specific. As long as two participants have at least one PDP and EDP in common, they can exchange the required discovery information.

For the sake of interoperability, every DDS-RTPS implementation must support at least the SDP. This protocol consist of the Simple Participant Discovery Protocol (SPDP) as PDP protocol, and the Simple Endpoint Discovery Protocol (SEDP) as EDP protocol.

Figure 2.4 depicts the typical SDP sequence diagram. It consists of the following phases:

Bootstrapping: SDP discovery starts using a list of known hosts. This list contains the locators (typically unicast or multicast Internet Protocol (IP) addresses) for which a participant will announce its presence. Alternatively, if there are no specified IP addresses, default addresses will be used. Both options can be used together.

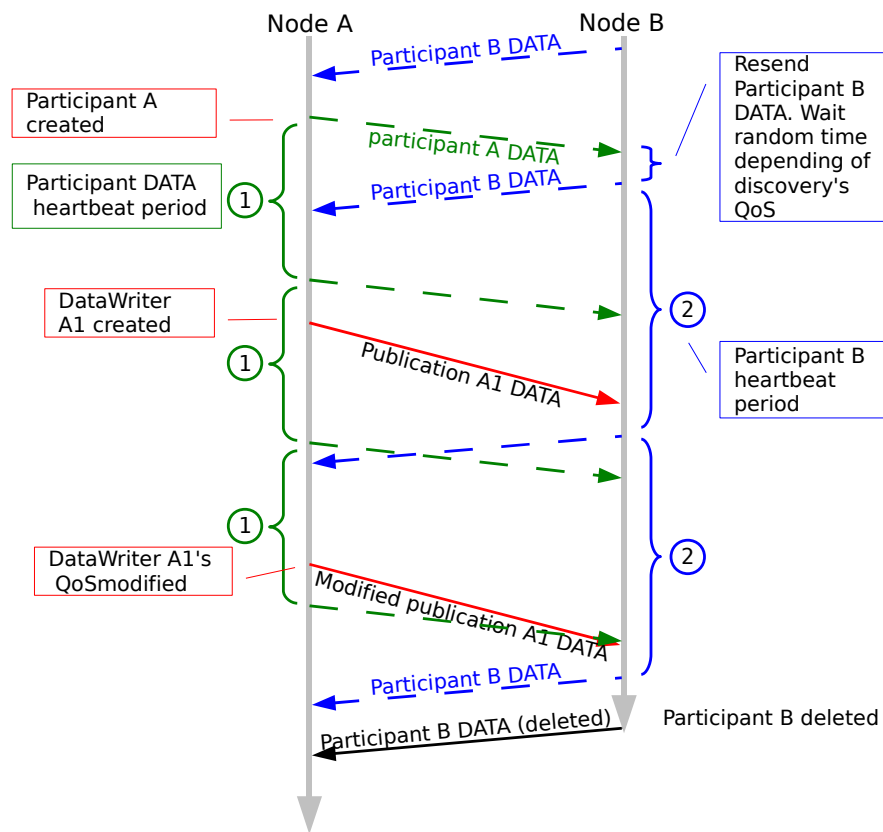


Figure 2.4: SDP sequence diagram.

Participant Announcement: Using the list of nodes obtained during bootstrapping, participants start the process of discovering other participants. This discovery, restricted to participants in the same DDS domain, is done via SPDP.

In SPDP, nodes periodically send a special message called *SPDPdiscoveredParticipantData* or simply *Participant DATA*. If multicast is available, each participant sends a unique *Participant DATA* message.

When a participant receives a *Participant DATA* message, it sends back another *Participant DATA* with its own information. Then, the participant stores the received participant information locally.

Participant DATA contains information for establishing communication between two participants, that is, information related to the protocol version, vendor identification, unicast and multicast locators (transport protocol to use, IP address and port combinations), and information about how to track participant liveness. Also, the information specifies what EDPs the remote participant supports.

Endpoint Announcement: Once a participant checks that the remote participant

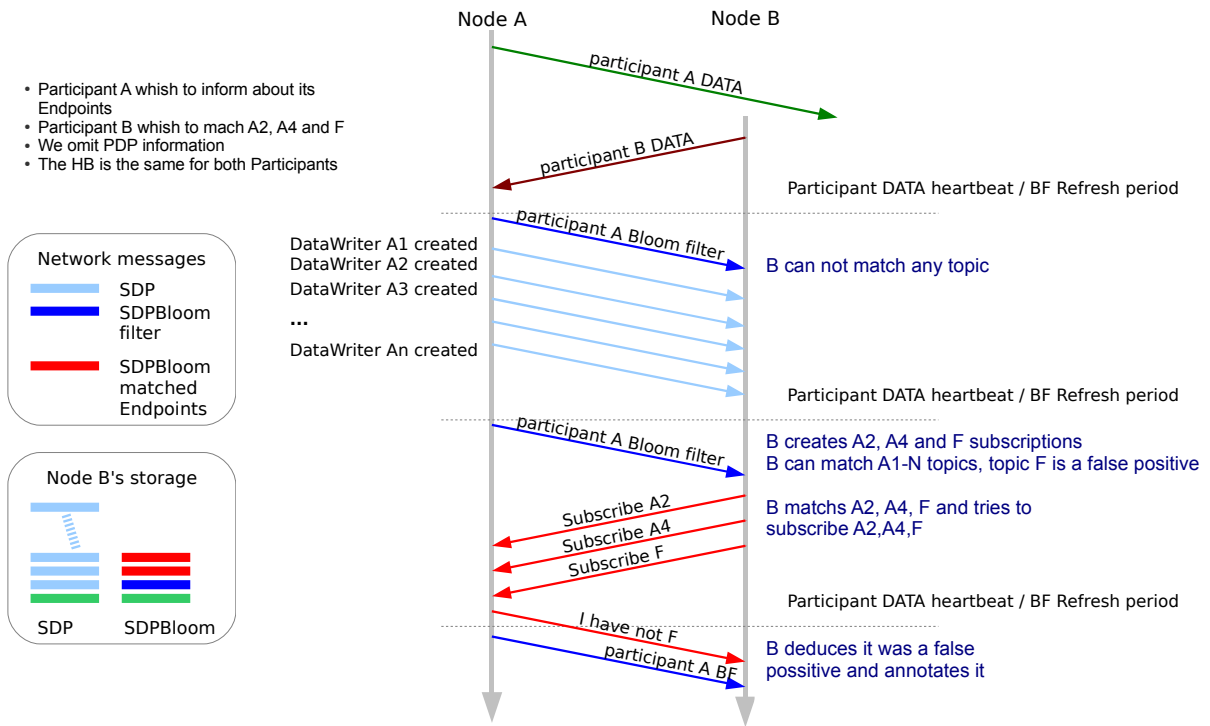


Figure 2.5: SDP-Bloom sequence diagram.

supports SEDP, they exchange endpoint information. Endpoint information consists of a topic name, a topic type, and a set of QoS parameters. Using this information, each participant checks if local and remote endpoints are compatible, and if so, they start a publish-subscribe communication.

2.2.3 SDP Bloom

In this section we introduce SDP-Bloom, an alternative SDP that helps to understand the flexibility and potential of the DDS-RTPS specification.

Due to the pure distributed nature of SDP, each participant stores information about discovered participants, associated publications, and subscriptions in a local database. According to SEDP, each participant sends the description of its associated endpoints to every discovered remote participant. Therefore, each participant receives all the discovered participants' endpoint information, which can generate a high number of messages.

In this regard, in [92] we proposed a new discovery protocol for DDS, the SDP-Bloom. Figure 2.5 depicts the typical SDP-Bloom discovery sequence diagram. During PDP phase, participants using SDP-Bloom send a Bloom-filter summarizing

their associated endpoint information. This allows participants to avoid performing the EDP discovery phase when the received remote participant's Bloom filter does not match with any of the local endpoints.

In this way, SDP-Bloom takes advantage of using Bloom-filters for reducing the network overhead in terms of number of messages and total traffic, at the cost of a small false positive endpoint matching rate. For a more in depth study of SDP-Bloom functionality and its impact on DDS performance, please refer to [92].

2.2.4 Extensible Topic Types

As mentioned before, the DDS specification states that a topic has an associated data-type. However, binding a data-type with a publication:

- hinders system evolution and backward compatibility
- makes impossible to use *a priori* unknown data-types
- prevents from maintaining different data-types views

The OMG has recently standardized the **DDS-XTypes** specification [71]. This specification, among other features, relaxes the requirement of using compilation time static topic data-types.

The use of static topic data-types is simple, efficient, and provides compile-time type-safety. However, it requires types to be known *a priori* and compiled into the code. This characteristic prevents the implementation of generic bridging services for DDS, such as the one proposed in Chapter 3. In this regard, the main functionality of a bridging service is to subscribe to certain data-content in a data-space and to publish the received data-content to a different data-space. The use of static topic data-types requires developers to declare the data-types that the bridging service will use for each data-space. Consequently, it is not possible to implement a generic DDS bridging service (*i.e.*, one that accepts topics of any data-type) by using static topic data-types.

In DDS, type schemas –that is, the names and definitions of a type and its fields– are represented by *TypeObjects*. A *TypeObject* consists of a *TypeObjectKind* and a list of members.

In addition to using locally serialized *TypeObjects*, the DDS-XTypes specification states that a serialized form of these (called *DataRepresentation*) must also be published (and therefore can be received) automatically during DDS discovery.

This mechanism allows applications to dynamically discover topic data-types. Once the types have been discovered, participants can use the *DDS Dynamic Topic Types* API (also defined in the DDS-XTypes specification) to publish and to subscribe to topics of types unknown at compilation time.

Since the OMG has recently published the final version of the specification, some of the results presented in this Thesis (in particular, the ones presented in Chapter 3) are based on the current (and preliminary) version of the specification.

2.3 Session Initiation Protocol (SIP)

SIP [83] is an IETF [39] protocol specification. SIP was originally specified within the Multiparty MULTimedia SessIon Control working group (MMUSIC WG) [33], and later within the Session Initiation Protocol working group (SIP WG) [34] and the Session Initiation Protocol Project INvestiGation working group (SIPPING WG) [35]. Currently, the Session Initiation Protocol Core working group (SIPCore WG) [37] is maintaining and continuing the development of the core SIP specifications.

SIP is an application-layer signaling protocol for creating, modifying, and terminating sessions with one or more participants. These sessions usually include VoIP calls, multimedia streaming, and multimedia conferences. SIP also provides a registration function that allows users to upload their current location.

In this section we study the fundamentals of SIP, which is one of the protocols our work is based on. In this regard, in Chapter 4 we propose a protocol for session signaling in data-centric publish-subscribe environments; and in Chapter 5 we propose an extension to RELOAD, which is a specification originally [44] conceived as an evolution of SIP.

2.3.1 Overview of SIP functionality

SIP is an application-layer signaling protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls or videoconferences, among others. In this regard, SIP supports sdp, a protocol for describing the details of the session, such as the type of media, codec, or sampling rate.

SIP dynamically manages sessions –such as multicast conferences– that is, it allows applications to invite participants to already existing sessions. In addition, it allows to negotiate and add (or remove) new media and formats to an existing session.

SIP transparently supports name mapping and redirection services, which en-

ables personal mobility. The SIP architecture –originally designed with a client-server model– also defines application servers to make the session highly configurable.

SIP supports five facets of establishing and terminating multimedia communications:

- **User location:** determination of the end system to be used for communication.
- **User availability:** determination of the willingness of the called party to engage in communications.
- **User capabilities:** determination of the media and media parameters to be used.
- **Session setup:** "ringing", establishment of session parameters at both called and calling party.
- **Session management:** including transfer and termination of sessions, modifying session parameters, and invoking services.

From the previous facets, in next sections we focus on user location, session setup, and session management; which are the facets this Thesis is focused on.

2.3.2 Basic concepts of SIP

The following terms have special significance for SIP [83]:

- **Address-of-Record:** A SIP or SIPS Uniform Resource Identifier (URI) that points to a domain with a location service. It can map the URI to another URI where the user might be available. Typically, the location service is populated through registrations. An address-of-record is frequently thought of as the "public address" of the user.
- **Message:** Data sent between SIP elements as part of the protocol. SIP messages are either requests or responses.
- **Method:** The method is the primary function that a request is meant to invoke on a server. The method is carried in the request message itself. INVITE and BYE are SIP method examples.
- **Request:** A SIP message sent from a client to a server, for the purpose of invoking a particular operation.

- **Response:** A SIP message sent from a server to a client, for indicating the status of a request sent from the client to the server.
- **Session:** From the sdp specification [40]: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session".
- **SIP URI and SIPS URI:** A SIP URI is a URI that identifies a communications resource. SIPS URIs use the same syntax than SIP URIs, but they specify that the resource must be contacted securely using Transport Layer Security (TLS) [6].
- **User Agent Client (UAC):** A logical entity that creates a request, and then uses the client transaction state machinery to send it. The role of UAC lasts only for the duration of that transaction. In other words, if a piece of software initiates a request, it acts as a UAC for the duration of that transaction. If it receives a request later, it assumes the role of a User Agent Server (UAS) for the processing of that transaction.
- **User Agent Server (UAS):** A logical entity that generates a response to a SIP request. The response accepts, rejects, or redirects the request. This role lasts only for the duration of that transaction. In other words, if a piece of software responds to a request, it acts as a UAS for the duration of that transaction. If it generates a request later, it assumes the role of a UAC for the processing of that transaction.
- **User Agent (UA):** A logical entity that can act as both a UAC and UAS.

2.3.3 User location

SIP offers a discovery capability [83]. In this regard, whenever a user wants to initiate a multimedia session, SIP discovers the current host(s) at which the target user is reachable. This is similar to the DDS discovery previously studied, where DDS entities (instead of users) discover compatible entities for exchanging DDS data-samples. In this regard, DDS application developers can take advantage of SIP location service for performing DDS discovery as we will see in Chapter 6.

According to the Request For Comments (RFC) [83], the discovery process is frequently accomplished by SIP network elements, such as proxy servers and redirect servers, which are responsible for receiving a request, determining where to send it –based on knowledge of the location of the user– and then sending it there.

To do this, SIP network elements consult an abstract service known as location service, which provides address bindings for a particular domain. These address

bindings map an incoming address-of-record (e.g., sip:bob@biloxi.com) to one or more URIs that are somehow closer to the desired user (e.g., sip:bob@engineering.biloxi.com). Ultimately, a proxy will consult a location service that maps a received URI to the UA(s) at which the desired recipient is currently residing.

Registration creates bindings in a location service for a particular domain. A registration associates an address-of-record URI with one or more contact addresses. Thus, when a proxy for that domain receives a request whose Request-URI matches the address-of-record, the proxy will forward the request to the contact addresses registered to that address-of-record.

There are many ways for establishing the contents of the location service. One way is administratively. In the above example, an existing corporate database is the source of the information about Bob. However, SIP provides a mechanism for a UA to create a binding explicitly. This mechanism is referred to as registration.

Registration entails sending a REGISTER request to a special type of UAS known as a registrar. A registrar acts as the front end to the location service for a domain, reading and writing mappings based on the contents of REGISTER requests. The location service is then typically consulted by a proxy server that is responsible for routing requests for that domain.

Figure 2.6 illustrates the overall registration process, where a user named *carol* registers her location. Note that the registrar and proxy server are logical roles that can be played by a single device in a network; for the sake of clarity the two are separated in this illustration. Also note that UAs may send requests through a proxy server in order to reach a registrar if the two are separate elements. In the figure we can also observe how the proxy sip.chicago.com requests for *carol*'s location upon *bob*'s INVITE request.

SIP does not mandate a particular mechanism for implementing the location service. The only requirements are that a registrar for some domain must be able to read and write data to the location service, and a proxy or redirect server for that domain must be capable of reading those data.

A registrar may be co-located with a particular SIP proxy server for the same domain.

The location service is just an abstract concept. It generally contains information that allows a proxy to input a URI and receive a set of zero or more URIs. This information indicates the proxy where to send the request.

Registrations are one possible way to create this location information, but not the only one. Alternatively, the administrator can also configure arbitrary mapping functions at his discretion.

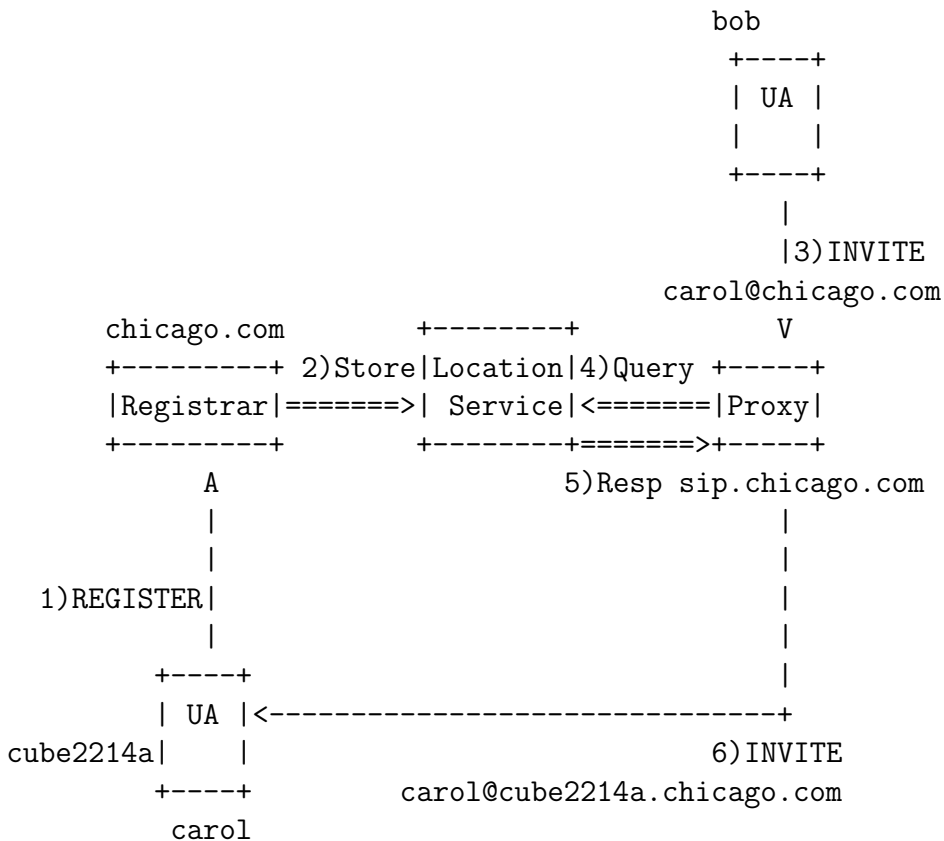


Figure 2.6: SIP registration and invitation example.

In addition, SIP registrations are not limited to only one device per user. For instance, a single user can register to the same URI both his SIP phone at home and the one in the office.

2.3.4 Session setup and management

According to the RFC [83], whenever a UAC desires to initiate a session (for example, audio, video, or a game) it formulates an INVITE request (see Figure 2.7).

The INVITE request asks a server for establishing a session. This request may be forwarded by proxies, eventually arriving at one or more UAS that can potentially accept the invitation.

If the UAS accepts the session, it will send a 2xx response (which means success) back to the UAC. If the invitation is not accepted, the UAS will send a 3xx, 4xx, 5xx or 6xx response (which respectively mean redirection, client error, server error,

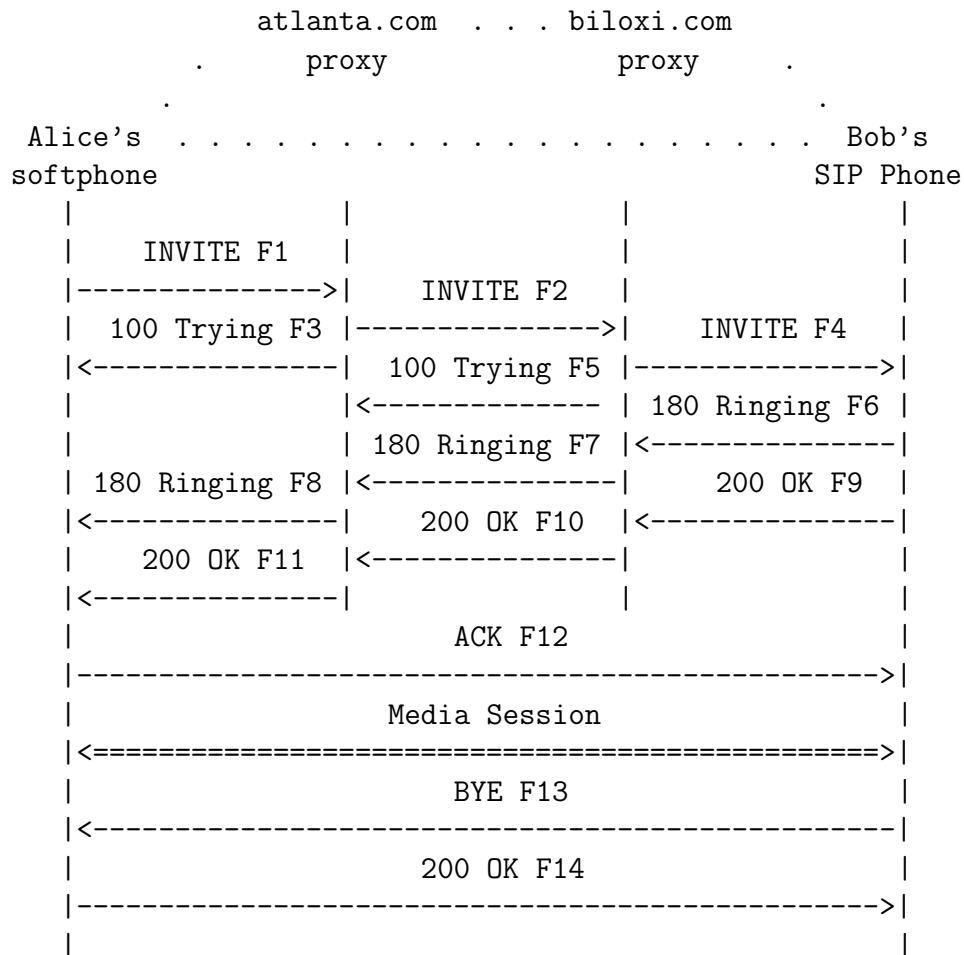


Figure 2.7: SIP invitation dialog.

and global failure). For more information about SIP responses, please refer to the RFC [83].

After possibly receiving one or more provisional responses, the UAC will get one or more 2xx responses or one non-2xx final response. The the UAC sends an ACK for every final response it receives. A 2xx response to an INVITE establishes a session. It also creates a dialog between the UA that issued the INVITE and the UA that generated the 2xx response. Therefore, when a UA receives multiple 2xx responses from different remote UAs (because the INVITE forked), each one of these 2xx responses establishes a different dialog. However, all these dialogs are part of the same call.

A UA that supports INVITE must also support ACK, CANCEL, and BYE; as they are closely related to the INVITE request.

ACK confirms the reception of a 2xx response. CANCEL requests for the cancellation of an INVITE request. BYE requests for the finalization of an active session.

SIP also provides mechanisms for modifying an existing session. The modification can involve changing addresses or ports, adding a media stream, deleting a media stream, and so on. This is accomplished by sending a new INVITE request within the same dialog that established the session. An INVITE request sent within an existing dialog is known as a re-INVITE.

2.4 REsource LOcation And Discovery (RELOAD)

RELOAD [44] is an IETF protocol for building and maintaining P2P systems on the Internet. It provides a generic, self-organizing overlay network service, allowing nodes to efficiently route messages to other nodes and to efficiently store and retrieve data in the overlay.

The working group responsible of the RELOAD specification is the Peer-to-Peer Session Initiation Protocol working group (P2PSIP WG) [36]. The original name of RELOAD was Peer-to-Peer Session Initiation Protocol (P2P-SIP).

P2P-SIP was initially conceived as an evolution of SIP that replaced the relatively fixed hierarchy of SIP servers with a P2P overlay network [9]. The main objective of P2P-SIP was to increase the scalability of SIP by taking advantage of using a P2P approach instead of a client-server one. In this regard, the authors of [66] propose a mechanism for eliminating the need for the sign-up process and the servers for this purpose.

RELOAD's original usage was focused on providing user registration, call setup, and instant messaging [99] [8]. In subsequent versions of the specification RELOAD was generalized in order to provide a P2P-based resource location and discovery service to any application (instead of only supporting the ones based on SIP).

2.4.1 Main features

RELOAD provides several features that are critical for a successful P2P protocol for the Internet [44] :

- **Security framework:** P2P networks are often established among untrusted peers. RELOAD leverages a central enrollment server to provide credentials for each peer which can then be used to authenticate each operation. These credentials consist of a certificate assigned to each node that joins the overlay.

- **NAT Traversal:** RELOAD is designed to function in environments where many (if not most) of the nodes are behind NATs or firewalls. RELOAD uses ICE [84] for providing NAT traversal functionality.

- **High Performance Routing:** RELOAD has been defined with a simple, light-weight forwarding header, thus minimizing the amount of effort required by intermediate peers.

- **Pluggable Overlay Algorithms:** For the sake of extensibility, RELOAD defines an abstract interface to the overlay layer to simplify implementing a variety of structured and unstructured overlay algorithms. Additionally, RELOAD uses a Chord-based Distributed Hash Table (DHT) [102] [103] as its default overlay algorithm.

- **Usage Model:** RELOAD is designed to support a variety of (existent and future) applications. RELOAD allows the definition of new application usages, each of which can define its own data types, along with the rules for their use. In RELOAD, a Kind is the definition of a data type and its accessing rules. Therefore, new usages must define the Kinds they store in the RELOAD overlay.

These properties allow RELOAD to adapt to different scenarios, while providing efficient and secure resource discovery.

2.4.2 Architecture and types of nodes

A RELOAD overlay instance consists of a set of nodes arranged in a partly connected graph. Each node in the overlay is assigned a numeric Node-ID for the lifetime of the node which determines –together with the specific overlay algorithm in use– its position in the graph and the set of nodes it connects to. The Node-ID is also tightly coupled to a certificate. Figure 2.8 shows a trivial example of RELOAD overlay included in [44].

RELOAD defines two different types of nodes:

- **Peer:** A host that is participating in the overlay. Peers are responsible for holding some portion of the data that has been stored in the overlay and also route messages on behalf of other hosts as required by the overlay algorithm.

- **Client:** A host that is able to store data in and retrieve data from the overlay but does not perform routing or data storage for the overlay.

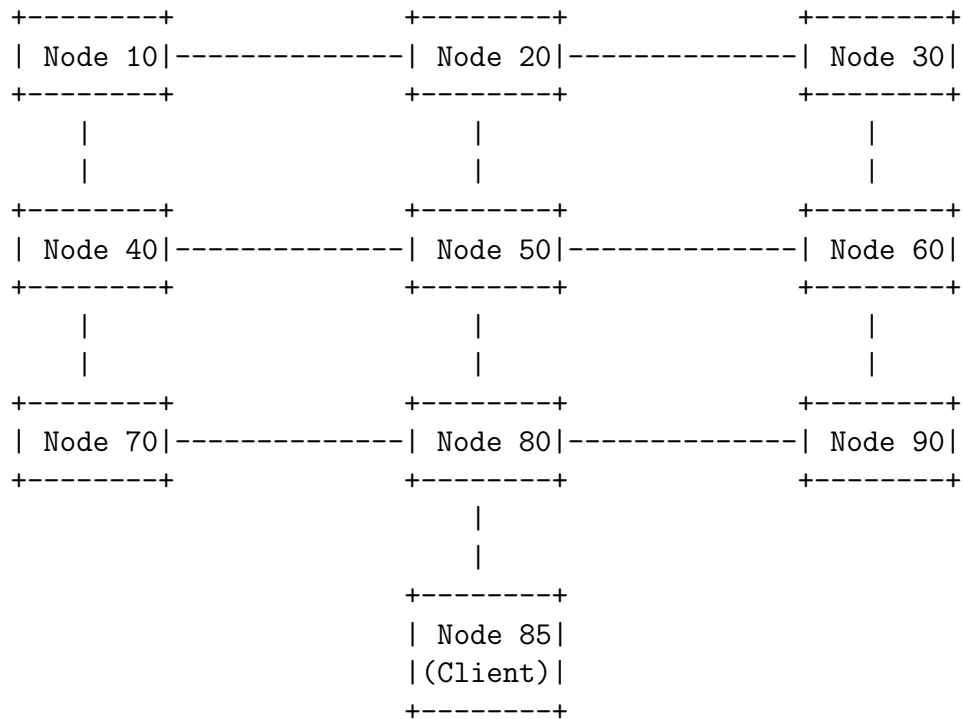


Figure 2.8: Trivial example of RELOAD overlay.

2.4.3 Operations

As a P2P protocol, RELOAD defines a set of operations for joining, leaving, and maintaining the overlay. There are also operations for storing and retrieving data.

In addition, RELOAD defines two new operations for interacting with other overlay nodes: Attach and AppAttach.

Attach allows nodes to establish a direct TCP or User Datagram Protocol (UDP) connection to other overlay nodes. This connection is a direct channel between two Nodes exchanging RELOAD messages.

AppAttach is similar to Attach, but in this case nodes use the generated connection for exchanging application data (and not RELOAD traffic). In this sense, one of the parameters of AppAttach is the Application-ID, which identifies connection's target application.

RELOAD nodes send both Attach and AppAttach requests using the destination Node-ID (and not the IP address). In this sense, RELOAD allows nodes for establishing direct connections to any member of the overlay. This is feasible even if the target node is behind a NAT.

Resource-ID	
Kind 1	Kind 2
Value	Value
Value	Value
Value	

Figure 2.9: Example of data stored in RELOAD.

2.4.4 Data storage

RELOAD references the location of the data stored in the overlay by using an identifier which is known as Resource-ID, and its format depends on the particular DHT algorithm the overlay uses.

Each location in the overlay may contain data elements corresponding to multiple Kinds. In addition, a resource location may contain multiple elements of a given Kind. Each Kind is identified by a Kind-ID, which is a code point either assigned by Internet Assigned Numbers Authority (IANA) or allocated out of a private range. Figure 2.9 depicts an example of resource location [44].

The RELOAD specification defines the following data models for storing data, though developers can define their own structures as part of a new RELOAD usage:

- **single value:** There can be at most one item in the set and any value overwrites the previous item.
- **array:** Many values can be stored and addressed by a numeric index.
- **dictionary:** The values stored are indexed by a key. Often this key is one of the values from the certificate of the peer sending the Store request.

2.4.5 Usages

RELOAD's original usage was focused in providing a rendezvous service for SIP. In this way, communicating peers use RELOAD for establishing a connection, and then they perform the usual SIP negotiation [43].

There are other RELOAD usages like [42], which allows federating different SIP domains; or [27], which enables for managing the RELOAD overlay using Simple Network Management Protocol (SNMP). However, there is still no usage for supporting DDS.

2.5 Constrained Application Protocol (CoAP)

CoAP [97] is an ongoing IETF protocol specification. The working group responsible of its specification is the Constrained RESTful Environments working group (CoRE WG) [38].

According to the current IETF draft [97], CoAP is a specialized web transfer protocol for use with constrained nodes and constrained (*i.e.*, low-power, lossy) networks. CoAP provides a request-response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types.

CoAP can be used not only between nodes on the same constrained network, but also between constrained nodes and nodes on the Internet. The latter is possible since CoAP can be translated to HyperText Transfer Protocol (HTTP) for integration with the web.

Application areas of CoAP include different forms of Machine-to-Machine (M2M) communication, such as home automation, construction, health care or transportation. Areas with heavy use of sensor and actuator devices that monitor and interact with the surrounding environment.

Although CoAP is conceived for using a request-response interaction model, [29] specifies a simple protocol extension for CoAP that enables CoAP clients to subscribe to certain resources. Once a client subscribes to a particular resource, the corresponding server will send notifications upon changes on the subscribed resource.

DDS data-space Interconnection Service (DDS-IS)

IN this chapter, we consider the problem of communicating disjoint data-spaces that may use different schemas to refer to similar information. In this regard, we propose a DDS interconnection service capable of bridging DDS domains as well as adapting between different data schemas. These features constitute a first step for deploying DDS in large-scale deployments.

A key benefit of our approach is that is compliant with the latest OMG specifications, thus the proposed service does not require any modifications to DDS applications.

The chapter identifies the requirements for DDS data-spaces interconnection, presents an architecture that responds to those requirements, and concludes with experimental results gathered on our prototype implementation.

3.1 Motivation

DDS was originally designed for isolated LANs in which data sources (publishers) and data consumers (subscribers) are located in the same geographical location.

Problems arise if a subscriber has interest in data being originated in a different data-space, especially if that data-space is separated by a bandwidth-constrained network. This is the case of a company that relies on DDS for delivering some data-

content internally, and needs to share a subset of that data-content with another company.

A simplistic solution to the previous use case would be to merge the two data-spaces. That is, to implement a service that publishes and announces all the topics from one data-space to the other. However, the direct merging of the data-spaces would create some significant problems.

The first problem is related to scalability. Every publication available in one data-space will be indiscriminately announced in the other data-space. This may be wasteful in resources in situations where there are no consumers to that information on the second data-space.

Other problems are concerned to data compatibility. For instance, each data-space may have its own (possibly different) data model (*i.e.*, data names, types, and/or definitions). These models could be incompatible even if they refer to the same data items. For example, temperature data might be expressed in Celsius in one data-space and in Fahrenheit in the other one. Therefore, in order to keep the end-to-end application logic unchanged, data should be automatically transformed by the publish-subscribe infrastructure when bridging data-spaces with different data models. As an additional benefit, this will ease the integration between new and legacy DDS applications.

Access control and information confidentiality adds another dimension to this problem: Some parts of the data should be confined in a data-space whereas other data should be public.

Finally, another relevant issue is related to the possibility of having different delivery requirements for the information that flows between and within the different interconnected data-spaces. For example, system administrators should be able to establish the QoS requirements that domains must meet in order to be bridged.

In this chapter we propose a novel solution for inter-domain entity and data-type signaling, and data-content exchange in DDS environments.

Our solution, named *DDS-IS*, is fully compatible with the current DDS standard specification and allows for advanced content-centric operations, like content-based filtering and data transformations.

DDS-IS establishes a content-aware interconnection service between different data-spaces with the following distinctive features:

a) **Scalability.** The overall traffic load is drastically reduced in comparison with direct data-space merging;

b) **Data model compatibility and confidentiality.** The seamless communication between data-spaces with dissimilar data models (topics of different names or data-

types) was not possible so far. DDS-IS solves this issue by properly transforming and/or isolating topic samples, if necessary.

c) **Delivery guarantees.** DDS-IS establishes QoS requirements for bridged data-spaces;

d) **Standards compliance.** The design of DDS-IS is fully compliant with the OMG DDS specification, hence it is implementation agnostic.

e) **Performance.** Conducted experiments demonstrate that the impact of the proposed service on communications performance is well within the acceptable limits for most real-world uses of DDS.

The remainder of the chapter is organized as follows. In Section 3.2 we present a motivating use case scenario for the proposed service. In Section 3.3 we identify the service requirements. In Section 3.4 we describe the proposed DDS-IS architecture. In Section 3.5 we study the interactions among DDS-IS architectural elements. In Section 3.6 we include relevant implementation details. In Section 3.7 we address the conducted experiments and analyze the obtained results. In Section 3.8 we conduct a study of the impact of the service in DDS QoS policies. Finally, in Section 3.9 we summarize the main conclusions of our work and discuss some open issues that are treated in next chapters.

3.2 Motivating use case example

DDS is a middleware for solving information sharing and data-exchange on real-time distributed systems. In this sense, DDS-based solutions are often deployed in industrial and mission critical applications. Two problems associated to these deployments are system heterogeneity and scalability.

As systems grow in complexity, incompatibility problems arise among different software versions or among applications of different providers. In the same way, the natural evolution of enterprises generates scalability problems, and it is necessary to support bigger scenarios, perhaps even deployed among distant locations.

From these problems, it emerges the need of defining DDS bridging service that improves DDS scalability and eases the interoperation of heterogeneous DDS-based applications.

To identify the requirements of this service, we now detail a motivating scenario that requires the functionality of a DDS bridging service.

Figure 3.1 includes an example deployment scenario for the DDS-IS. Specifically, this example shows how DDS-IS can be used in an olive oil factory. Although

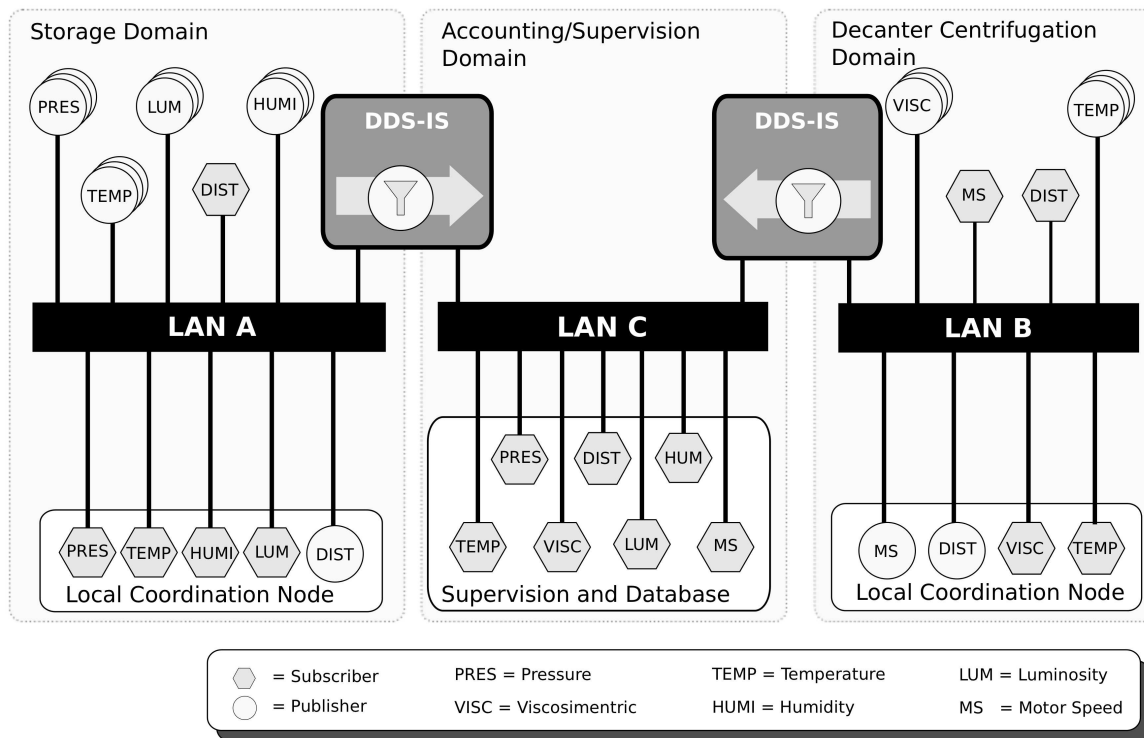


Figure 3.1: Example deployment scenario.

other scenarios are possible, this example illustrates the functionalities of the proposed service.

The olive oil factory in our example generates data-content in two different environments, namely centrifugation system and storage system. In addition, there is a third environment (accounting and general supervision system) that stores the most relevant data-updates.

Decanter centrifugation is the part of olive oil production that separates olive oil from vegetation water and solid materials. In order to produce high quality oil, operators must control the centrifugation speed, centrifuge temperature, and oil density. The centrifugation system of our example has three centrifuges. Each one publishes viscosimetric information and temperature information (with a ratio of one update per second). Each centrifuge is also subscribed to two topics: one for controlling centrifugation speed, and other for controlling the input and output valves. A local coordination node controls the decanter centrifugation system, by subscribing to temperature and viscosimetric topics, and controlling centrifuge speed and valves.

Storage is the part of olive oil production where the oil is stored until its bottling and distribution. In order to conserve the quality of the oil, the system must monitor

the pressure, temperature, humidity and light exposure of the storing tanks. In our example, we have assumed three oil tanks. Each one of these tanks publishes pressure, temperature, humidity, and light exposure information (with a rate of one update per second), and subscribes to a valve control topic (*i.e.*, three publishers per topic for a total of 14 publishers, and three subscribers). A local coordination node controls the valve status, and monitors published metrics (using four different subscribers and one publisher). Local coordination node can react to received values if necessary (for example, by activating a refrigeration system).

Although the generated data-content is mainly useful within its associated system (centrifugation or storage), the administrators of this factory want to store the most relevant events of each system in a central data-base, generating alerts if certain trigger values are reached. However, administrators do not want to modify either the current sensor deployment or the local coordination nodes logic. Moreover, it is desirable to keep the data-content of each system isolated (in order to avoid overflowing local networks with unnecessary information).

Figure 3.1 presents a solution to the described scenario requirements. In particular, this solution is based on deploying two DDS-IS nodes: one for bridging and filtering data-content from the storage system (*i.e.*, from LAN A to LAN C), and the other for bridging and filtering data-content from the centrifugation system (*i.e.*, from LAN B to LAN C). Using this deployment, DDS-IS nodes bridge only relevant data-samples to the accounting and general supervision system, while they avoid overflowing storage and centrifugation systems with unnecessary updates.

3.3 Requirements

From the previous example, we identify the following requirements that are not currently covered by DDS, and which the proposed service must fulfill:

R1: *Domain bridging*. DDS-IS must connect different DDS data-spaces. These data-spaces are groups of publishers and subscribers that exchange data using compatible DDS middleware implementations and transport configuration.

R2: *Topic bridging*. DDS-IS must connect different DDS topics. A topic has associated a data-type. Consequently, in order to bridge different topics our service must be able to adapt data-samples between different data-schemas.

R3: *Type-based transformations*. In order to adapt data-samples between different data-schemas, DDS-IS must be able to perform type-based data transformations. As an additional requirement, the service must allow users for defining their own transformations.

R4: *Content-based filtering*. DDS-IS must be able to use samples' content for determining if a sample should be bridged, or silently dropped such that it does not flow through the bridge.

R5: *QoS policies consistence*. DDS-IS must be compatible with existent DDS QoS profiles. DDS-IS must allow administrators for establishing what QoS requirements domains must met in order to be bridged. The service must ensure that the QoS policies of the interconnected entities are fully compatible with the ones configured for the DDS-IS. In addition, DDS-IS must allow administrators for integrating data-spaces with different (perhaps even incompatible) sets of QoS.

R6: *Transparency*. DDS-IS must not require modifications to existing DDS applications. This feature will enable legacy and new DDS applications to communicate through the proposed service. In this sense, our service will be an interoperability solution for connecting dissimilar data-spaces.

R7: *Standards compliance*. DDS-IS must be compliant with DDS standards family. This will ensure the interoperability of the service with any DDS-compliant middleware implementation.

R8: *Performance and scalability*. DDS-IS must have a low impact on DDS applications performance, mainly in terms of latency overhead. In addition, the system should improve DDS deployments scalability in terms of overall network traffic load.

R9: *Information confidentiality*. DDS-IS must provide mechanisms for filtering out sensitive information from bridged data-samples. Specifically, DDS-IS must be able to use data-samples' content for determining what data-samples should be discarded (this functionality is covered by R4). DDS-IS must also be able to selectively filter specific data-sample fields (instead of the whole sample) using the functionality covered by R3.

In the following section we describe the architecture of our system, which satisfies the identified requirements.

3.4 Internal architecture

DDS-IS architecture adopts a layered design based on the following motivations. First, it increases system modularity, by allowing the specific implementation of each layer to be changed without impacting the rest of the layers.

Second, it allows existing applications to remain independent of new proposed DDS-IS entities while benefiting from the service.

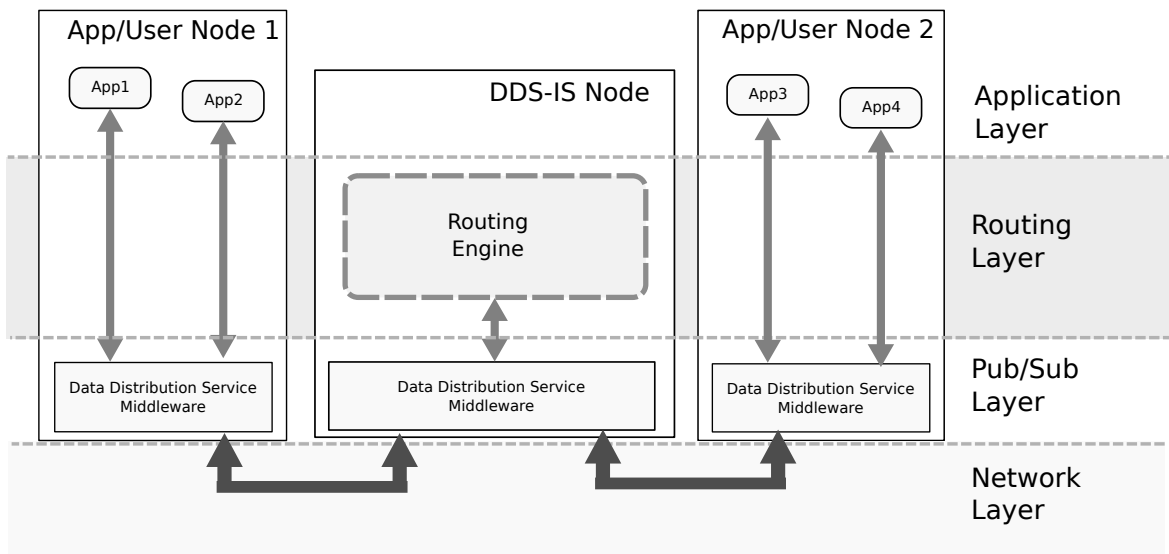


Figure 3.2: Layered DDS-IS architecture.

And third, it simplifies the overall understanding of the system, because it classifies entities of different functionality in different layers.

Figure 3.2 depicts the architecture of our solution. It can be seen as composed of a set of components located on different nodes. The components of each node are arranged in four layers, namely, application, routing, pub/sub, and network layer. In the following subsections we explain the functionality of each layer and its contained components.

3.4.1 Application layer

The application layer is the topmost layer of our architecture, and it is populated by DDS applications. These applications exchange information through DDS data-spaces by means of the underlying DDS middleware.

In our design, entities on this layer do not communicate directly with the routing layer. Instead, applications exchange data through the pub/sub layer using the existing standard DDS API (*i.e.*, in the same way as if the DDS-IS was not present). This feature allows applications to remain independent of the DDS-IS, and therefore they do not require any changes.

DDS-IS operation is totally transparent to the application layer. From the perspective of application layer entities, the only noticeable change is the increase in the data-content that they can access. This is because the DDS-IS enables the inter-operation of applications associated with different data-spaces (*i.e.*, different DDS

domains), virtually combining the content of the involved data-spaces.

The DDS-IS also enables applications with different data models to exchange data-content. This is provided by the ability of the DDS-IS to transform data on the fly. Again, these transformations are independent of the application layer, and therefore existing applications require no modifications for benefiting from the interconnection service.

As we have stated before, our solution does not modify the interactions between applications and DDS data-spaces. This is also valid for DDS compatibility checks. In this sense, if a DW (*e.g.* an application DW) and a DR (*e.g.* a DDS-IS DR) determine that they are not fully compatible (by performing the necessary DDS discovery-checks), they will not start exchanging data-content. Consequently, in order to bridge two applications with different data-types, or associated to different data-spaces, the DWs and DRs of the DDS-IS have to be properly configured.

In the same way, the QoS profiles associated to bridged entities must be compatible with the QoS profiles configured for the DDS-IS. As we will discuss later in Section 3.6.5, DDS-IS administrators can configure the service for enforcing the same QoS configuration for the two interconnected data-spaces. Alternatively, system administrators can configure DDS-IS for integrating two domains with different (perhaps even incompatible) sets of QoS policies.

3.4.2 Routing layer

The routing layer is the core of our design. This new layer extends the functionality of the DDS specification: it allows for bridging DDS information between different DDS domains (*i.e.*, between DDS data-spaces). It enables information flow across different DDS topics, optionally transforming and filtering the exchanged data.

As a result, from the application layer perspective, two different (perhaps even remote) data-spaces appear as a single (and extended) DDS data-space. This functionality is provided by the Routing Engine (see Figure 3.3), which contains the following components:

Discovery Monitor: This component processes the discovery events generated by the underlying DDS middleware (*i.e.*, by B-DRs). Whenever a change occurs in the DDS discovery information, the DDS middleware notifies the Discovery Monitor. The Discovery Monitor then retrieves the received discovery information from B-DRs and delivers it to the proper Domain Route.

Data Engine: This component controls one or more DRs and DWs in the pub/-sub layer. The Data Engine has two missions. First, it takes received data samples from the pub/sub layer, and pushes those samples to the proper Topic Route. Sec-

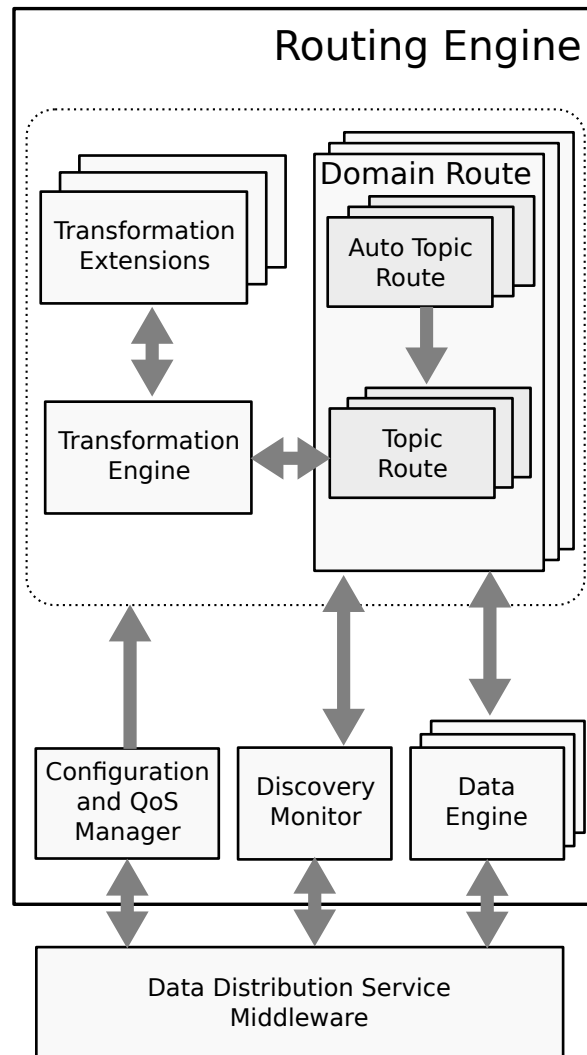


Figure 3.3: DDS-IS Routing Engine.

ond, it delivers resulting samples (from Topic Routes) to DWs so that they can be published. There is one Data Engine associated with each Domain Route.

Domain Route: A mapping between two different DDS domains (*i.e.*, data-spaces). A Domain Route contains multiple Auto Topic Routes and Topic Routes. The Domain Route interacts with the Discovery Monitor in order to obtain discovery information from a set of B-DRs in the pub/sub layer. After receiving discovery updates from the Discovery Monitor, the Domain Route triggers the creation of DDS and DDS-IS entities according to the service configuration. A DDS-IS instance can contain multiple Domain Route instances, each bridging a pair of DDS domains.

Topic Route: A mapping between an input topic (*i.e.*, the topic from where the DDS-IS receives data) and an output topic (*i.e.*, the topic where the DDS-IS pub-

lishes data). Therefore, a Topic Route is defined by an input topic name and type, and an output topic name and type. Additionally, a Topic Route can apply transformations to data-samples using the Transformation Engine (explained below). The behavior of a Topic Route is as follows: first, it receives data from its associated Data Engine; then, it applies transformations (if any); and finally, it delivers processed data to the Data Engine for publication. A Domain Route instance can contain multiple Topic Route instances, each one bridging a pair of DDS topics.

Transformation Engine: This element receives input data samples from a Topic Route, applies a set of transformations to those samples, and returns the resulting data samples. If a Topic Route requests for multiple transformations, the Transformation Engine processes them in sequence and following a predefined order (this is important because the output of a transformation and the input of the subsequent transformation must be type-compatible). Supported transformations are defined through multiple Transformation Extensions.

Transformation Extension: This element receives input data samples, applies a transformation to those samples, and returns the result of the transformation. To be able to process data of any type, Transformation Extensions use *DDS Dynamic Topic Types*. In order to increase system flexibility, A DDS-IS instance can contain multiple Transformation Extension instances, each one describing a particular data transformation. Transformation Engine can load external Transformation Extensions implemented by DDS-IS users.

Auto Topic Route: A generic Topic Route that bridges a range of topics. The topic name and type for bridged topics need not be known *a priori*, and therefore Auto Topic Route provides auto-configurable topic bridging. Auto Topic Routes are configured with a subset of topic names and/or topic types for which the automatic setup is allowed. A Domain Route instance can contain multiple Auto Topic Route instances, each one covering a different set of topics.

Configuration and QoS Manager: Following the philosophy of the DDS standard, our design relies on a set of QoS policies for configuring the behavior of routing layer elements. Indeed, some of these QoS policies also control the configuration of entities belonging to the DDS core layer (for example, Topic Route QoS profiles also control the configuration of the associated DW and DR). The Configuration and QoS Manager is the component that controls the proper configuration of the DDS-IS.

3.4.3 Pub/Sub and network layers

The pub/sub layer (see Figure 3.2) contains the entities associated to the DDS middleware (described in Section 2.2.1), and interacts with system's upper layers using

the standard DDS API.

The pub/sub layer allows the DDS-IS to access to DDS data-spaces. In order to provide this functionality, the pub/sub layer perform the following tasks:

- (1) it notifies the routing layer about changes in discovery information,
- (2) it receives new data samples and delivers these samples to the routing layer, and
- (3) it publishes data obtained from the routing layer to DDS data-spaces.

In addition to the entities already introduced in Section 2.2.1, this layer includes two DDS WaitSets. A DDS WaitSet is an standard DDS entity that can be used to wait for specific DDS Events. DDS WaitSets are awoken whenever certain pre-defined trigger conditions are met. Specifically, our system uses the following two DDS WaitSets:

Discovery WaitSet: This element receives updates from B-DRs (see Section 2.2.1). It notifies the Discovery Monitor in the routing layer when a change occurs to discovery. This allows the DDS-IS to react to the appearance and disappearance of entities to/from the connected data-spaces.

Data WaitSet: This element notifies its associated Data Engine in the routing layer about the reception of new data samples in one of its DRs. This allows the DDS-IS to process DDS samples from the DDS core layer as soon as they are received.

The network layer is the lowest layer of the proposed architecture (see Figure 3.2). It represents the platform-dependent components that are necessary for exchanging data-samples (such as the operating system, computer hardware, or communication network). One of the advantages of using a network middleware (such as DDS) is that it decouples the application logic from the particularities of the network layer. Consequently, we do not need to define any requirements for this layer, except that the chosen DDS implementation must be DDS-RTPS-compliant.

3.5 System operation

In order to illustrate the interactions among the previously described entities, in this section we provide sequence diagrams for the two main system use cases.

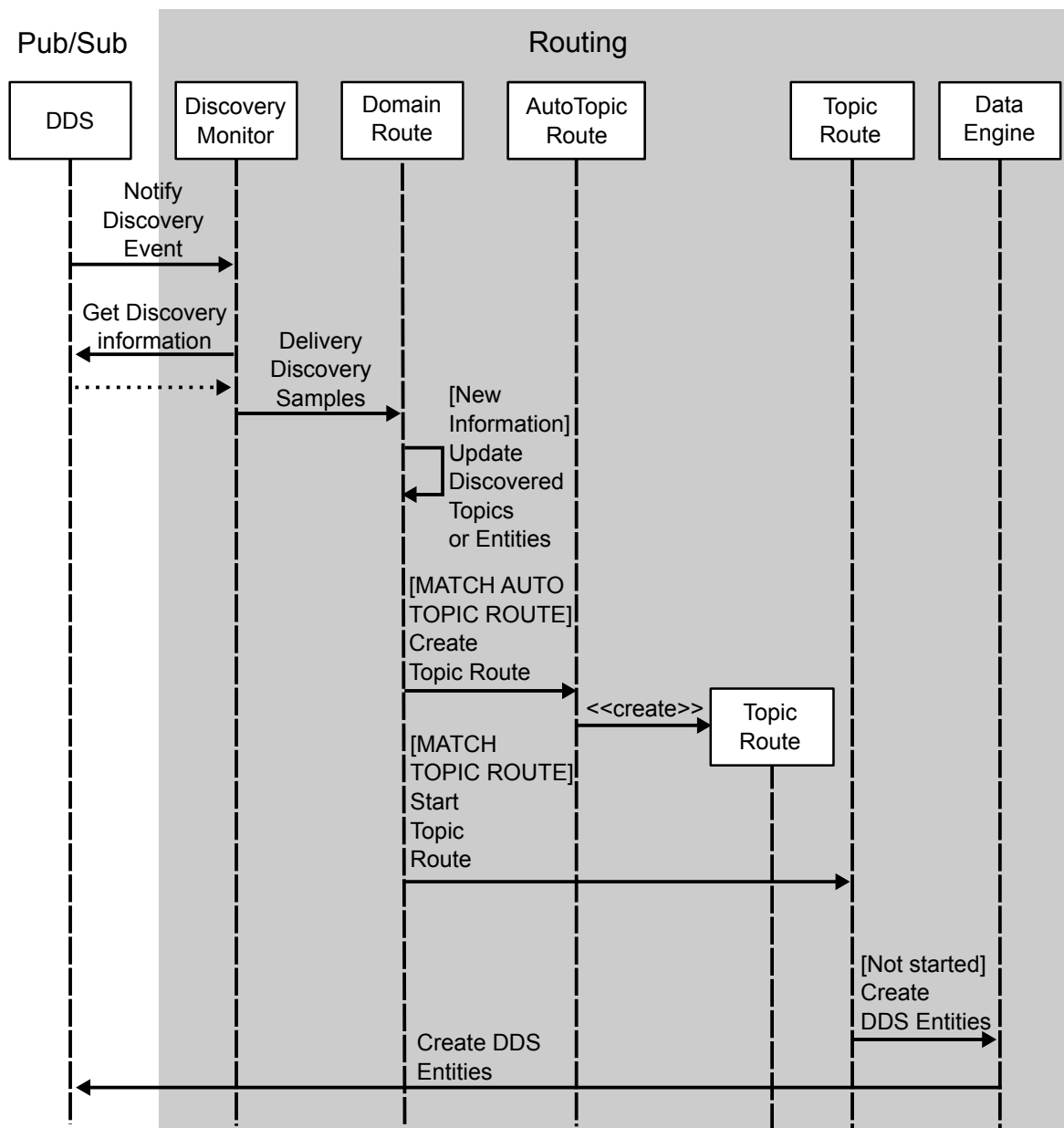


Figure 3.4: Discovery sequence diagram.

3.5.1 Discovery of new DDS entities

DDS discovery notifies DDS-IS about the creation or deletion of DDS entities in any of the interconnected domains. This information enables DDS-IS to react to the changes in the input and output data-spaces.

The sequence diagram of the discovery procedure is depicted in Figure 3.4. Specifically, this sequence diagram describes how the discovery of DDS entities trig-

gers the initialization of a Topic Route and all its associated entities.

A B-DR initiates the process when it receives a change in the discovery information of a DDS entity located in the same data-space (but associated to a different node). This event triggers a Discovery WaitSet (studied in Section 3.4.3), which then sends a notification to the Discovery Monitor.

Once the Discovery Monitor receives a discovery notification, it retrieves the nature of the originating B-DR (participant, publication or subscription), and the discovery information received by the B-DR.

At this point, the Discovery Monitor delivers all the received discovery samples to the Domain Route associated to the originating B-DR. Received samples can contain multiple changes in the discovery, which are processed separately. These changes can refer either to the creation or removal of remote DDS entities.

Using the received information, the Domain Route updates an internal register that stores information of previously discovered DDS entities and topics. This register enables the DDS-IS for determining if it is necessary to create new Topic Routes upon the reception of discovery information.

Then the Domain Route checks if any of its active Auto Topic Routes matches with the received discovery information. If necessary, the Domain Route creates new Topic Routes according to Auto Topic Routes' configuration.

Finally, the Domain Route checks the existing Topic Routes, and then determines if the discovered DDS entity is associated to any of them. If the Domain Route finds a matching Topic Route, the Domain Route will check if the Topic Route is already active. If the Topic Route is not active, the Domain Route starts the Topic Route (*i.e.*, the Topic Route requests the Data Engine to create the necessary DR and DW for performing topic bridging).

3.5.2 Data bridging

Data Bridging is the process of communicating data-samples between two different DDS data-spaces. If the input and output data-spaces share the same topic, the DDS-IS can bridge data-samples without performing any modifications to data-samples' structure.

However, the DDS-IS can also bridge data-samples between two data-spaces that do not share the same data-model. In this case, the DDS-IS must transform input data-samples for adapting them to the format required at the output data-space.

Figure 3.5 depicts the sequence diagram of the data bridging process. A DR (located in the pub/sub layer) initiates the process when it receives a new data-

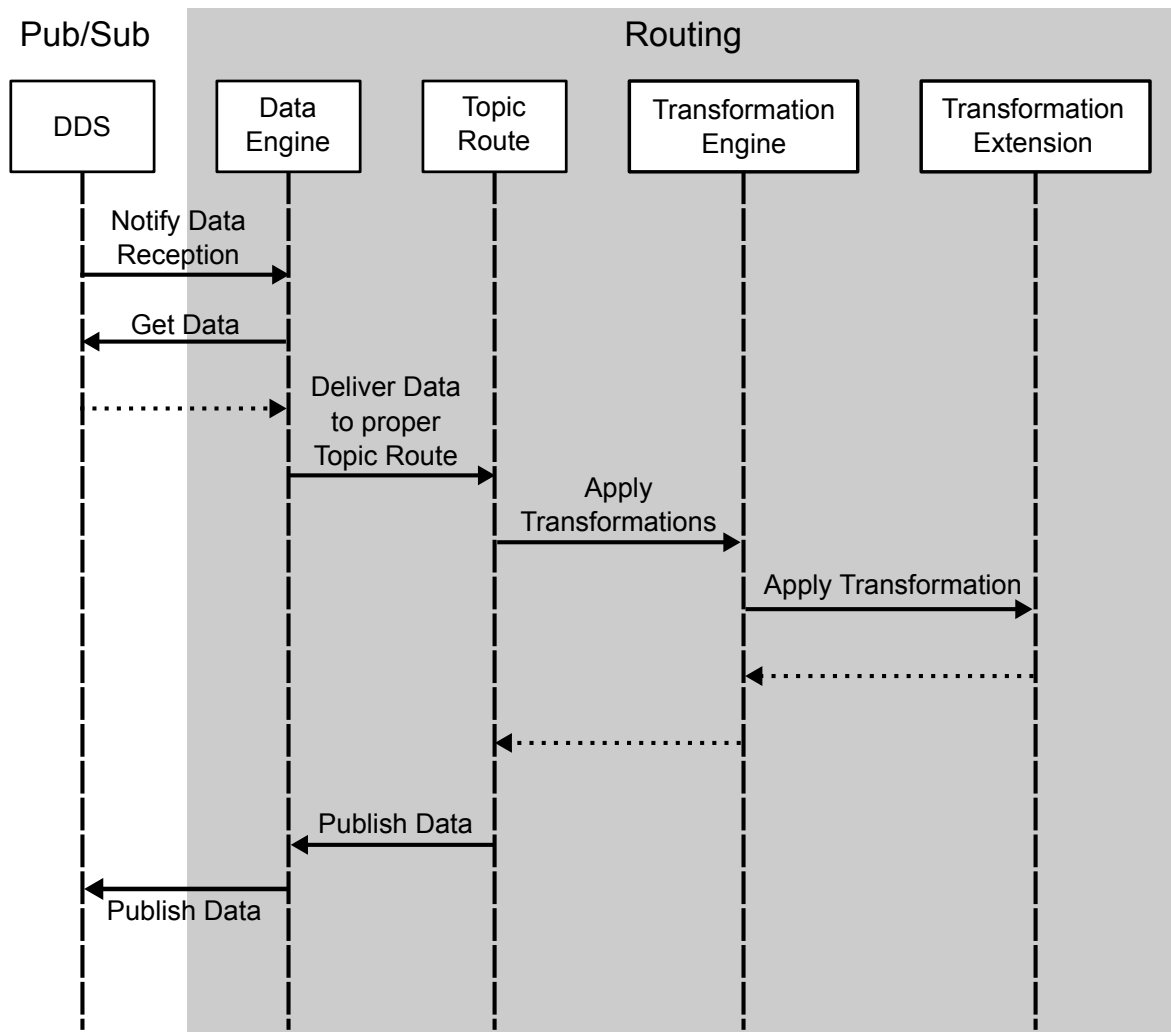


Figure 3.5: Data bridging sequence diagram.

sample from a DW located in the same data-space (but in a different node). This event triggers a Data WaitSet, also located in the pub/sub layer, which then sends a notification to the Data Engine associated to the DR.

After receiving the notification from DDS, the Data Engine retrieves all of the received samples from the originating DR, and delivers those samples to the Topic Route associated to that DR.

At this point, the Topic Route applies configured transformations (if any) to the input data-samples (using the Transformation Engine). Data Bridging then finishes with the publication of resulting data-samples to the output data-space by means of the proper Data Engine's DW (located in the pub/sub layer).

3.6 Implementation

After defining the DDS-IS architecture and its design, we have implemented a proof-of-concept prototype. This prototype was used to validate the proposed design, to perform field-testing, and to evaluate the performance impact of our service (for further information regarding conducted experiments and obtained results, please refer to Section 3.7). In this section, we explain the most relevant implementation decisions that we have taken during the prototype development.

3.6.1 Dynamic types

As we studied in Section 2.2.4, the DDS-XTypes API allows DDS applications to publish and subscribe topics with types unknown at the time the application was compiled.

Since DDS-IS needs to interact with DDS applications not known *a priori* (including DDS applications to be developed in the future), the DDS-IS makes an extensive use of the DDS-XTypes API in order to discover and manipulate previously unknown data types, including performing any necessary data transformations.

3.6.2 DDS middleware implementation

The DDS-IS prototype was implemented using Real Time Innovations Inc. (RTI)'s DDS version 4.4d [88]. We have chosen this implementation because it supports the *DDS Dynamic Topic Types* extension. This extension is a new API for the DDS standard family that was recently standardized by the OMG [70].

Nevertheless, our design is implementation-independent, as it only uses standardized DDS features plus the *DDS Dynamic Topic Types* extension. Consequently, the proposed design can be implemented using any other DDS-compliant implementation. Indeed, the implemented prototype is also compliant with the DDS-RTPS [69].

Regarding the used programming language and bindings, we have implemented our prototype in C, and we have compiled the source code using gcc version 4.4.3.

3.6.3 DDS signaling propagation

As we have mentioned, a Topic Route connects an input data-space's topic (*i.e.*, a set of application DWs) with an output data-space's topic (*i.e.*, a set of application

DRs). To enable the interoperation of these DWs and DRs, the DDS-IS must create both a DR at the input domain and a DW at the output domain.

In this scenario, the question naturally arises: when should the service create the input/output DR/DW? The answer to this question depends on the particular application requirements.

If system resources (memory and CPU) are scarce, DDS-IS should not create the paired DW-DR until a match between the input topic and the output topic is found (*i.e.*, both the matched DR and DW have been discovered).

However, if we need all of the DDS signaling information (*i.e.*, DDS discovery information) to propagate through the DDS-IS, the service should create both the DR and DW as soon as a matched publication or subscription is discovered.

Alternatively, it may be preferable for the system to set up Topic Route's entities as soon as the user configures a Topic Route (even if the DDS-IS does not find any matching DWs or DRs).

In order to support scenarios with different requirements and constraints, we decided that the prototype should allow the user to configure the triggering conditions for the creation of Topic Route's entities. In Section 3.6.6, we provide an example of how to configure the creation for the input and output of a Topic Route.

3.6.4 Propagating DataWriter information

By default, DDS middleware generates a unique Identifier (ID) (using hosting machine information and implementation specific parameters) for each DW. In the early stages of DDS-IS prototype implementation, this was also the behavior of the service. However, after some testing we concluded that this behavior prevents application DRs from distinguishing between original data-samples and replicas of those data-samples.

The ability of distinguishing between original data-samples and their replicas is especially critical for deployments with redundant DDS-IS nodes. In these scenarios, this functionality may significantly impact system resources consumption, given that it avoids the delivery of duplicated samples to final applications.

Consequently, we decide to enable the DDS-IS to maintain the original DW information, which is contained in input data-samples. The implemented prototype allows configuring if the original DW information is maintained, or if it is replaced by the DDS-IS DW information. In this way, the DDS-IS administrator is able to configure the system behavior according to the specific scenario requirements.

3.6.5 Quality of Service

As we described in Section 3.4.2, DDS-IS relies on DDS QoS profiles for configuring the behavior of the service. Consequently, DDS-IS also relies on DDS built-in compatibility-checking mechanisms for ensuring that QoS profiles of communicating entities are fully compatible.

DDS-IS QoS checking is performed on a hop-by-hop basis (*i.e.*, between DDS-IS entities and application entities located within the same domain). This feature allows system administrators for decoupling QoS for different domains, enabling the integration of heterogeneous (and perhaps even incompatible) scenarios. Of course, administrators can configure the same QoS policies for the two interconnected domains, which guarantees that QoS remain invariant across interconnected data-spaces.

Regarding QoS enforcement, DDS QoS policies were initially intended for working within single domains. Consequently, the behavior of DDS QoS policies in DDS-IS-based deployments is currently unknown. In this regard, in Section 3.8 we include an analysis of the impact of DDS-IS on each DDS QoS policy.

3.6.6 XML configuration format

The DDS-IS uses eXtensible Markup Language (XML) files to load its configuration. XML is a set of rules for encoding documents. These rules are defined in the XML 1.0 Specification produced by the W3C [106].

Figure 3.6 provides an example of a basic XML configuration file; specifically, it includes the following XML tags:

dds: The root of the XML document; it can also include the definition of QoS policies for the system.

dds_is: This tag includes the configuration of a DDS-IS instance. Therefore, every other DDS-IS should be included as a child of the `dds_is` tag.

domain_route: This contains the configuration for a Domain Route and must contain a `participant_1`, a `participant_2`, and a `session` tag.

participant_1 and **participant_2:** These tags define the input and output DDS domain for the service using the `domain_id` tag.

session: This contains the configuration for a set of Topic Routes and Auto Topic Routes that share the same Data Engine instance. Session tag can contain multiple `topic_route` and `auto_topic_route` tags.

```
<?xml version="1.0"?>
<dds xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../schema/dds_is.xsd">
  <dds_is>
    <domain_route name="DefaultDomainRoute" enabled="true">
      <participant_1>
        <domain_id>0</domain_id>
      </participant_1>
      <participant_2>
        <domain_id>1</domain_id>
      </participant_2>
      <session name="DefaultSession" enabled="true">
        <auto_topic_route name="All" enabled="true">
          <publish_with_original_info>
            true
          </publish_with_original_info>
          <publish_with_original_timestamp>
            true
          </publish_with_original_timestamp>
          <input participant="1">
            <creation_mode>IMMEDIATE</creation_mode>
          </input>
          <output>
            <creation_mode>IMMEDIATE</creation_mode>
          </output>
        </auto_topic_route>
      </session>
    </domain_route>
  </dds_is>
</dds>
```

Figure 3.6: XML configuration file example.

auto_topic_route and **topic_route**: The system uses these tags to set up the Auto Topic Routes and Topic Routes.

publish_with_original_info: This determines if the DDS-IS maintains the original DW information for publishing data to the output.

publish_with_original_timestamp: This determines if the DDS-IS maintains the original timestamp information for bridged data-samples.

input: This defines the configuration of the route input. It allows configuring filters for topic routing, the QoS policies for the input DR, and the `creation_mode`, which is defined below.

output: This defines the configuration of the route output. It allows configuring filters for topic discovery at the output, the QoS policies for the output DW, and the `creation_mode`, which is defined below.

creation_mode: This configures the way discovery information is propagated from the input to the output (or *vice versa*). In this way, when a DR is discovered at the output, the system can set up the corresponding Topic Route immediately (as in the example) or wait until a DW is discovered at the input.

3.7 Validation and performance evaluation

To study the operation and performance of the DDS-IS prototype, we conducted a set of experiments in a controlled LAN environment. Specifically, we measured the impact of the DDS-IS in terms of round-trip latency, throughput, and network traffic load.

3.7.1 Experimental set-up

The test-bed was composed of six Core i5 at 2.40GHz-2.66GHz machines (named lab01, lab02, lab03, lab04, lab05, and lab06) running Linux Kernel 2.6.32-22 x86_64 (Ubuntu 10.04) and the *RTI DDS 4.4d* middleware. These machines communicate through a 24-port Gigabit switch with Virtual Local Area Network (VLAN) support.

In our experiments, these nodes receive one of three possible roles:

Source-node: A node that generates DDS samples, receives responses for those samples, and calculates experimental statistics.

Echo-node: A node that receives DDS samples from source-nodes and generates responses.

Routing-node: A node that bridges two different networks.

We developed a benchmarking tool for conducting our experiments. This benchmarking tool consists of two components:

PerformanceTest_tester: This application runs in the source-nodes. It publishes samples into a topic called Ping and receives them from a subscribed topic, called Pong. Ping samples contain a sequence number, the publication timestamp, and a variable size payload as well. The structure complexity of Ping payload is variable. Pong samples only contain a sequence number and the original Ping publication timestamp. This timestamp is compared with the sample reception time for obtaining samples round trip time. Each Ping and Pong sample is sent in a separate message.

PerformanceTest_remote: This application runs in echo-nodes. It takes samples from the Ping topic and republishes their sequence number and timestamp field content into the Pong topic.

In the conducted tests, published samples performed a round trip between source-nodes and echo-nodes. This approach allows for the precise measurement of the sending and reception times without performing any clock synchronization. Consequently, at least two different routes have to be configured in the DDS-IS: the outgoing route (for bridging Ping samples) and the incoming one (for bridging Pong samples).

For all the reported evaluations, our benchmarking tool uses RTPS over UDP protocol, and sends each sample separately (*i.e.*, each UDP datagram contains only one RTPS sample).

Regarding sample sizes, Ping samples contain a payload whose size ranges from 256 to 8192 bytes and a protocol overhead of 118 bytes. Pong samples have the same protocol overhead as Ping samples, but their payload has a size of 12 bytes because they only contain a sequence number and a timestamp.

We considered the following metrics for the DDS-IS performance evaluation:

Throughput: The average number of samples per time interval (seconds) received from the data-space by subscribers.

Round-trip Latency: The average end-to-end elapsed time for delivering samples from the original source-node to the echo-node plus the elapsed time for delivering samples back to the original source-node. Our tool measures the latency at application level. In this sense, publishers measure the time before delivering data-samples to DDS middleware, whereas subscribers measure the time after DDS delivers data-samples to the benchmarking tool.

Generated traffic load: The total generated traffic (in bytes) for a specific net-

work segment in a particular scenario. This includes the payload plus all the overheads generated by the whole protocol stack.

In order to evaluate the proposed service performance, we conduct all the experiments in two different scenarios. Namely,

No-DDS-IS Scenario: Publishers and subscribers associated to source-nodes and echo-nodes were located in the same data-space, but in different networks. DDS entities communicate through one or more Layer-3 Linux Routers.

DDS-IS Scenario: Publishers and subscribers associated to source-nodes were located in a different data-space (also in a different network) than the ones associated to echo-nodes. DDS-IS interconnects the involved data-spaces.

In both scenarios, samples were published using the DDS Reliability QoS policy set to RELIABLE, which guarantees that the data sent by DWs is received by DRs.

Additionally, we have configured the QoS policy KEEP_HISTORY to ALL_HISTORY, which guarantees no sample is dropped from DDS buffers.

This configuration will force the DDS middleware to ensure the delivery of all the samples. However, it can cause that few samples have an extremely high latency, potentially biasing the average results. In this respect, the reported results only include the 95th percentile latency.

As part of the RELIABLE Reliability QoS configuration, DW send heartbeat messages to their DR. Each heartbeat message contains the sequence number of the most recent sample sent by the DW. We have set DDS DR heartbeats low-period (*i.e.*, the one used when DRs' cache has less than 5 samples) to 100 milliseconds, and we have set DDS DR heartbeats high-period (*i.e.*, the one used when DRs' cache has more than 15 samples) to 10 milliseconds.

We have also configured the DDS middleware for discovering DDS entities only in the interfaces of interest for each particular experiment. In this respect, all the experiments use a secondary network for launching the benchmarking tools without interfering on the tests.

As we explained in Section 3.6, the DDS-IS prototype has been implemented using RTI's C API. On the other hand, the benchmarking tool has been implemented using C++.

Regarding Linux kernel configuration parameters, we have used the default values associated to Ubuntu 10.04 for all the machines.

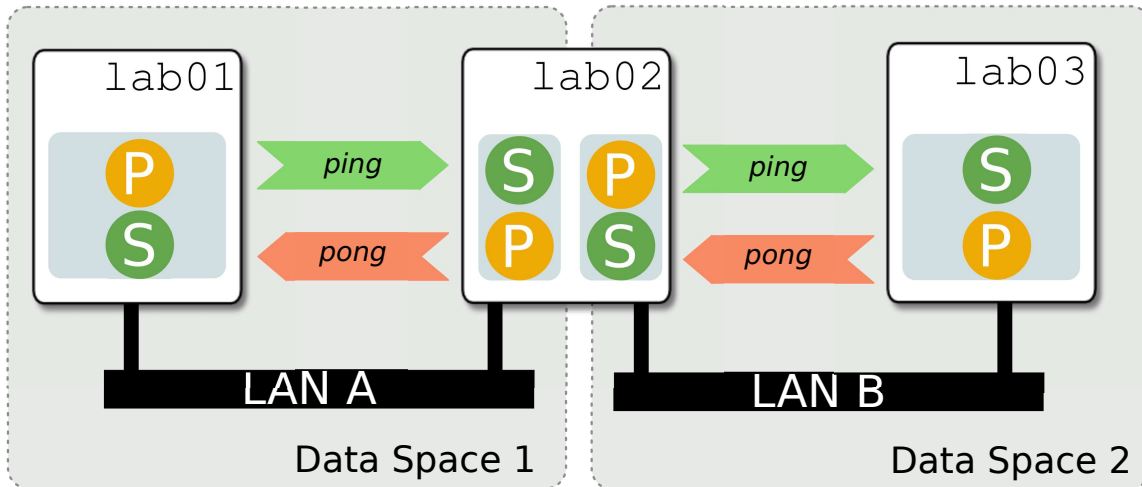


Figure 3.7: DDS-IS scenario for 1 to 1 performance experiments. In the N to N case, lab01 and lab03 run multiple publishers and subscribers.

3.7.2 DDS-IS Impact in N to N communications

In this section, we evaluate the impact of the DDS-IS in terms of round-trip latency and throughput for different payload (sample sizes), data types, and DDS-IS configurations when N publishers and N subscribers communicate following a 1 to 1 relationship.

To make the results comparable, both no-DDS-IS and DDS-IS scenarios used the same network topology: two LAN networks (A and B) connected through a computer (lab02) with two Ethernet interfaces.

In the no-DDS-IS scenario, lab02 is a single layer-3 Linux Kernel Router (we use just one data-space), whereas in the DDS-IS scenario, lab02 interconnects two different data-spaces and provides the proposed DDS-IS (see Figure 3.7).

Each test consisted of publishing 500,000 samples per topic and using a specific payload size. These samples were published at source-node (lab01) into the Ping topic; after being received by Ping subscriber at echo-node (lab03), they were published back using the Pong topic.

To avoid transitory behavior (warm-up) effects, before starting the tests we force some samples to be exchanged after the DDS entities discovery.

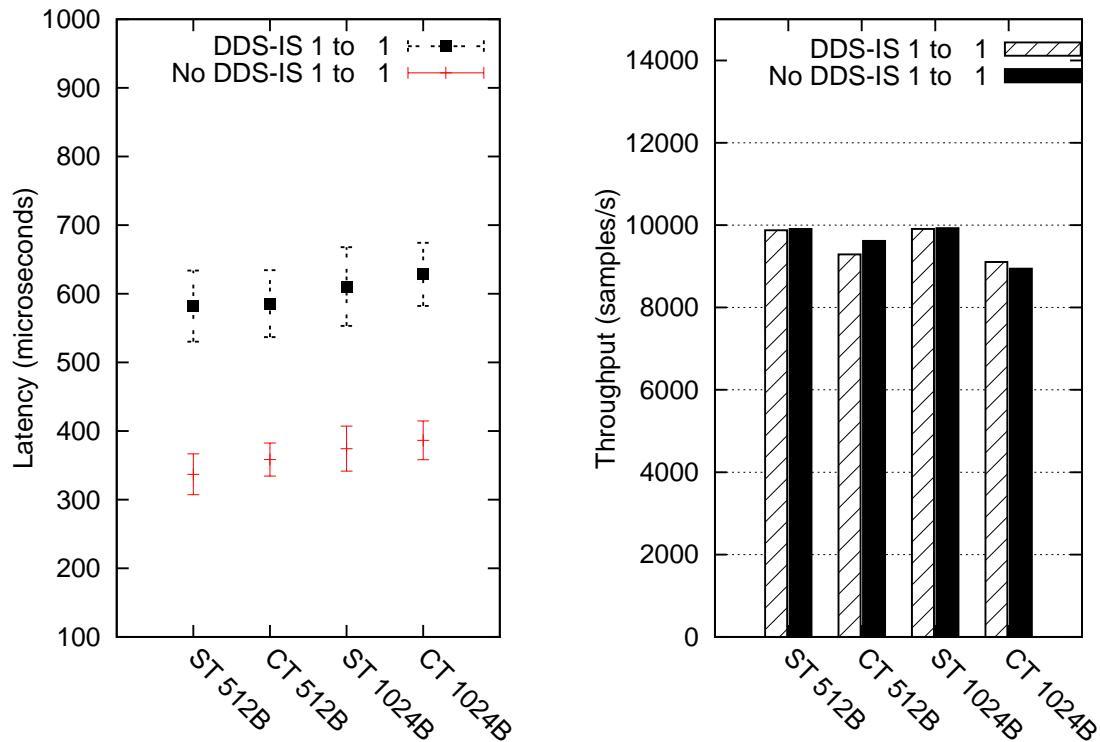


Figure 3.8: DDS-IS impact for different data types in 1 to 1 communications.

3.7.2.1 Performance impact in simple 1 to 1 communications

This first set of experiments aims at the evaluation of the performance impact for different data types and DDS-IS configurations when only one topic for Ping and one for Pong is active. Although the Pong topic was always the same, this experiment uses the following Ping topic types:

SimpleType (ST) 512, 1024 bytes: These contain a sequence of random bytes, a sequence number, and a timestamp.

ComplexType (CT) 512 bytes: This structure contains two levels of nesting. It includes a SimpleType (256 bytes) data, four 64byte-string fields (two of them in a nested struct), a long field, and eight Boolean fields.

ComplexType (CT) 1024 bytes: This structure contains three levels of nesting. It includes two ComplexType-512 byte structs.

Figure 3.8 shows the performance results of both no-DDS-IS and DDS-IS scenarios. In this experiment neither transformations nor filtering are enabled.

The results show that although the DDS-IS introduces approximately a round-trip latency overhead of 250 microseconds, it has no a significant impact on through-

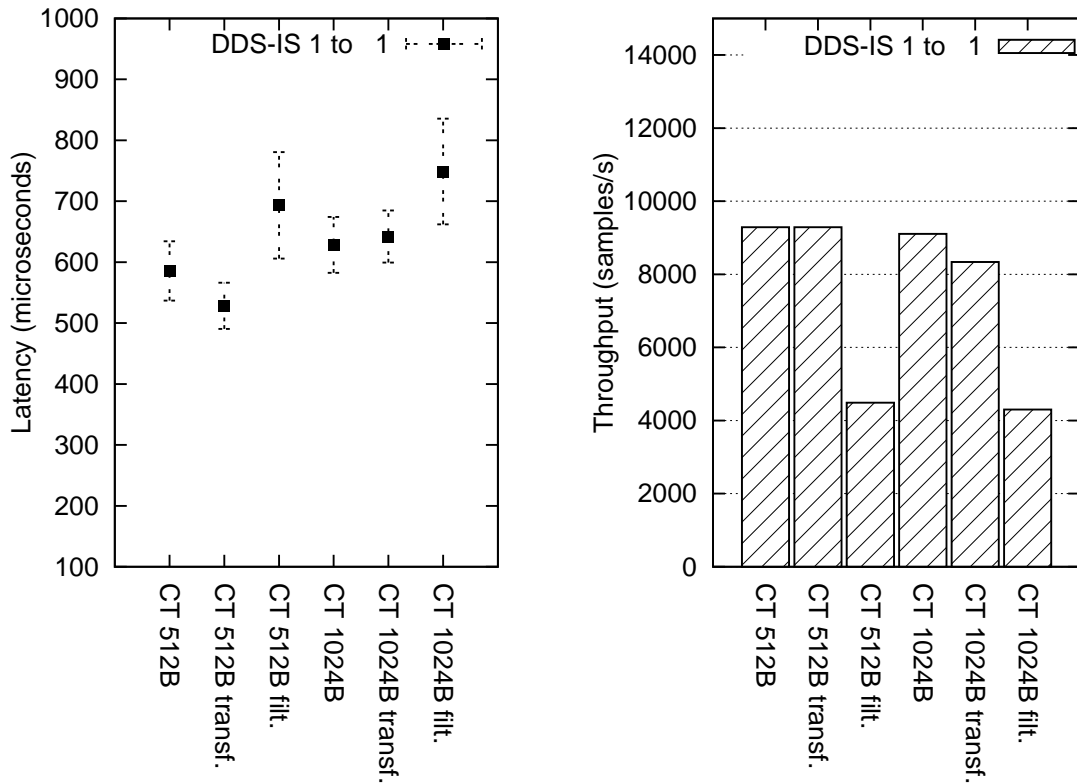


Figure 3.9: DDS-IS performance for filtering and transforming data.

put.

We can also conclude that in this case data complexity does not have a significant performance impact on DDS-IS scenario in comparison to no-DDS-IS scenario. Consequently, in the remainder experiments we focus our study on testing different sample sizes.

3.7.2.2 Performance impact of data transformation and filtering features

The DDS-IS allows loading external data transformations and applying these transformations to the output data. In this sense, the performance degradation is highly dependent on the complexity of the transformation. Therefore, rather than reporting an exhaustive analysis, this section aims to evaluate the cost of using simple data transformations, which will show the overhead introduced by inclusion of an additional function call to the datapath.

Figure 3.9 shows the incurred latency and throughput for a simple data trans-

formation definition (labeled as `CT size transf.`).

In particular, we have set up the DDS-IS for changing a long field and a Boolean field to a specific configured value. In this experiment, we observe that the complexity of the transformed structure degrades the throughput (see the obtained results for labels `CT 512B transf.` and `CT 1024B transf.`).

However, simple transformations have no significant negative impact on latency. Indeed, for `CT512` samples, using transformations reduces the average latency. This behavior is because the additional processing time introduced by transformations reduces the number of context switches in the routing-node Central Processing Unit (CPU), what reduces the average latency overhead.

To test impact of data filtering we configured a filter that drops samples according to the content of the samples. Published samples are labeled with a number randomly obtained from a uniform distribution for the interval 0 to 999. The DDS-IS filters the samples according to the following definitions. For `CT512` case (labeled in Figure 3.9 as `CT 512B filt.`) the filter was:

```
(long_cond_00 < 500) or (nested_ping.sequence_number = 4294967295)
```

Whereas for `CT1024` case (labeled in Figure 3.9 as `CT 1024B filt.`) it was:

```
(nested_complexping_02.long_cond_00 < 500)  
or (nested_complexping_01.nested_ping.sequence_number = 4294967295)
```

The first condition on each filter drops approximately the 50% of the samples. The second condition ensures the warm-up signaling samples traverse the DDS-IS filter.

According to the obtained results (see Figure 3.9), the filter introduces an average overhead of 110 microseconds when it uses a filtering condition on the first level of the type `CT512`, and 140 microseconds for evaluating the condition in a second level of nesting of the `CT1024`.

In addition, DDS-IS approximately halves the throughput compared to a scheme without filtering, as expected from the fact that this filter drops 50% of the samples.

This experiment validates the filtering and data transformation features of the DDS-IS. We can conclude that for the evaluated conditions DDS-IS accomplishes these tasks without a significant degradation of system performance.

3.7.2.3 Performance impact in N to N communications with multiple topics

In this section, we study how DDS-IS affects performance as the number of concurrent distinct topics (and Topic Routes) increases.

In this case, we use the same topology of Figure 3.7 but now we increase the number of bridged topics (up to 16). We always use an even number because our benchmarking tool utilizes two types of topics: Ping for outgoing samples and Pong for incoming ones.

This experiment is a worst-case scenario for the DDS-IS operation, as the capabilities of data multiplexing of DDS-IS are not used. Indeed, the DDS-IS has to maintain a pair of DR/DW per route.

More particularly, the last case studied in this section creates 16 DWs and 16 DRs in the same DDS-IS. Although this is not an appropriate scenario of DDS-IS operation (multiple 1-1 routes), these experiments provide some insight on the DDS-IS limits.

Figure 3.10 depicts the average round-trip latency as a function of the payload size. The results show that the DDS-IS performs reasonably well for samples with a size up to 6144 bytes when 2 to 16 concurrent routes (topics) are active. Specifically, the DDS-IS introduces an overhead from 250 to 400 microseconds for the round-trip latency.

For the 8192-sized payload, the scenarios with 8 and 16 concurrent topics reveal a performance degradation. In these cases the round-trip latency overhead introduced by the DDS-IS increases to 450 microseconds and 1.3 ms respectively. Additionally, the standard deviation for round-trip latency reaches 2 ms, which indicates the DDS-IS performance degrades for this load level.

Regarding throughput, the results in Figure 3.11 show that DDS-IS has an insignificant impact on the throughput for the scenarios with 2-4 concurrent topics.

For the 8 and 16 topics cases, the DDS-IS throughput achieves at most 30,000 samples per second (for the 64-byte payload), whereas the scenario without DDS-IS reaches 50,000 samples per second in the 8-topic case, and almost 75,000 samples per second in the 16-topic case (both for the 64-byte payload).

It is important to remark that in the 16-topic case the DDS-IS is publishing 30,000 samples in both directions (Ping samples from source-nodes to echo-nodes and Pong samples from echo-nodes to source-nodes). This is especially relevant in the 64-byte scenario, in which the size of Pong samples is similar to the size of Ping samples. In this scenario, while source-nodes and echo-nodes just manage one DW (for publishing Ping samples or for publishing Pong samples) and one DR (for receiving Pong samples or for receiving Ping samples), the routing-node has to manage a pair of DW/DR per topic.

We have also measured the impact on CPU consumption for the routing-node (*i.e.*, for lab02). Figure 3.12 shows that the maximum CPU consumption is obtained for the experiments using small packets, which are the ones with a highest rate of

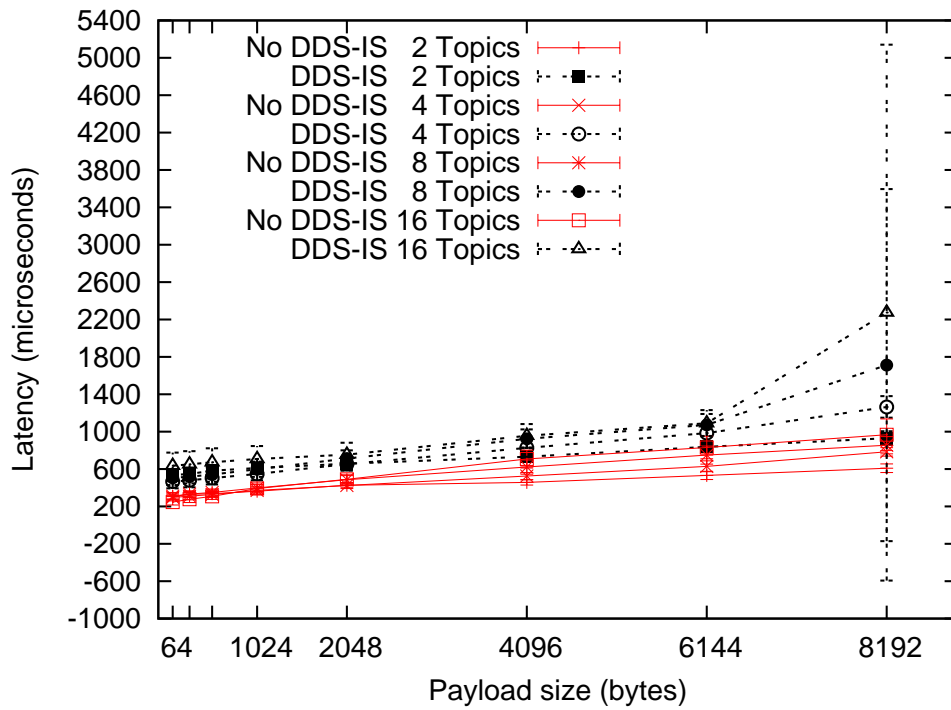


Figure 3.10: Average latency for different payloads and number of routes (topics) active.

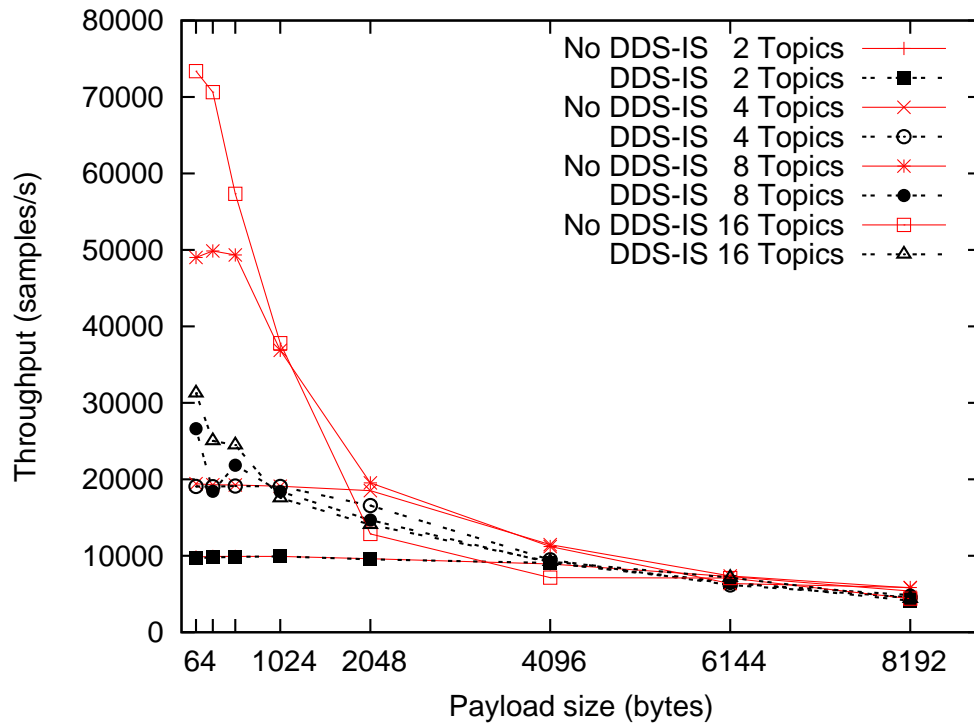


Figure 3.11: Total average throughput for different payloads and number of routes (topics) active.

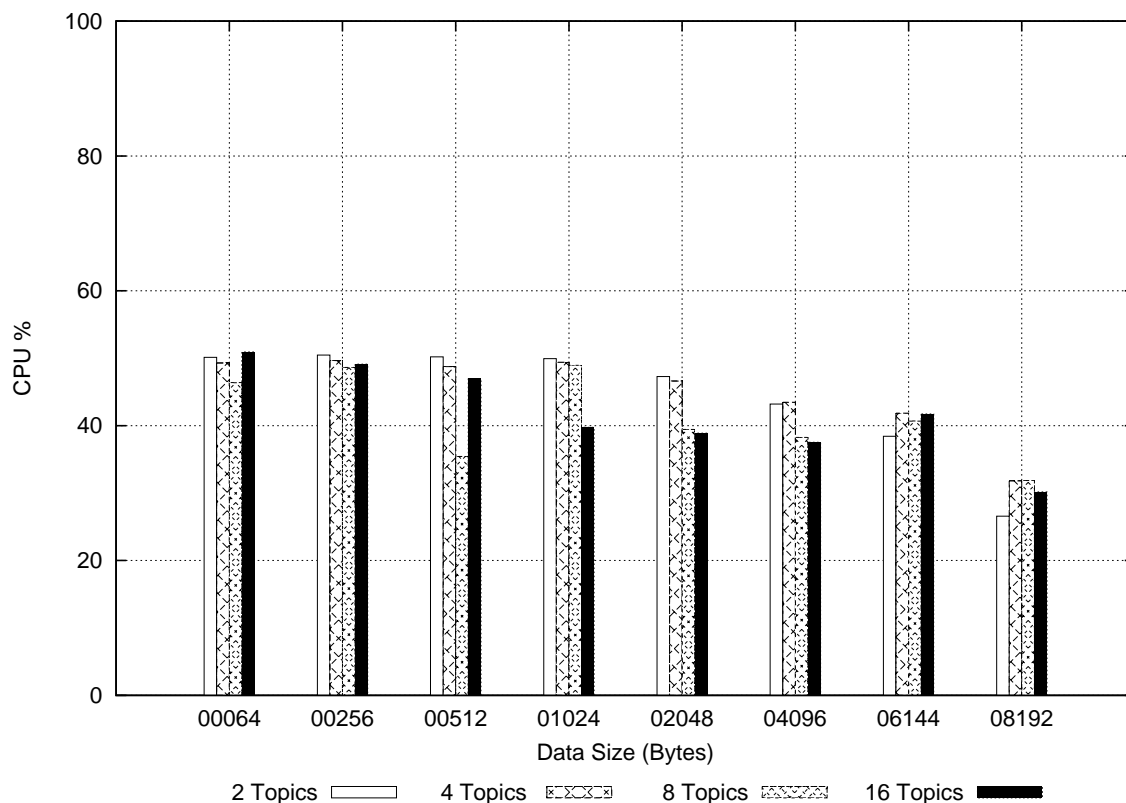


Figure 3.12: CPU impact for different payloads and number of routes (topics) active.

samples per second. The obtained results also show that the degradation in performance for bigger sizes is not due to CPU saturation. Instead, it is mainly due to synchronization problems between the middleware and the used Ethernet interfaces, along with the increased cost of packet disassembling, assembling, and loss-recovery.

3.7.3 DDS-IS impact in M to N communications

In the previous section, we studied the limits of the DDS-IS in a scenario where no data multiplexing is present. The following experiments analyze the performance of the DDS-IS when it delivers several copies of certain published data to multiple subscribers.

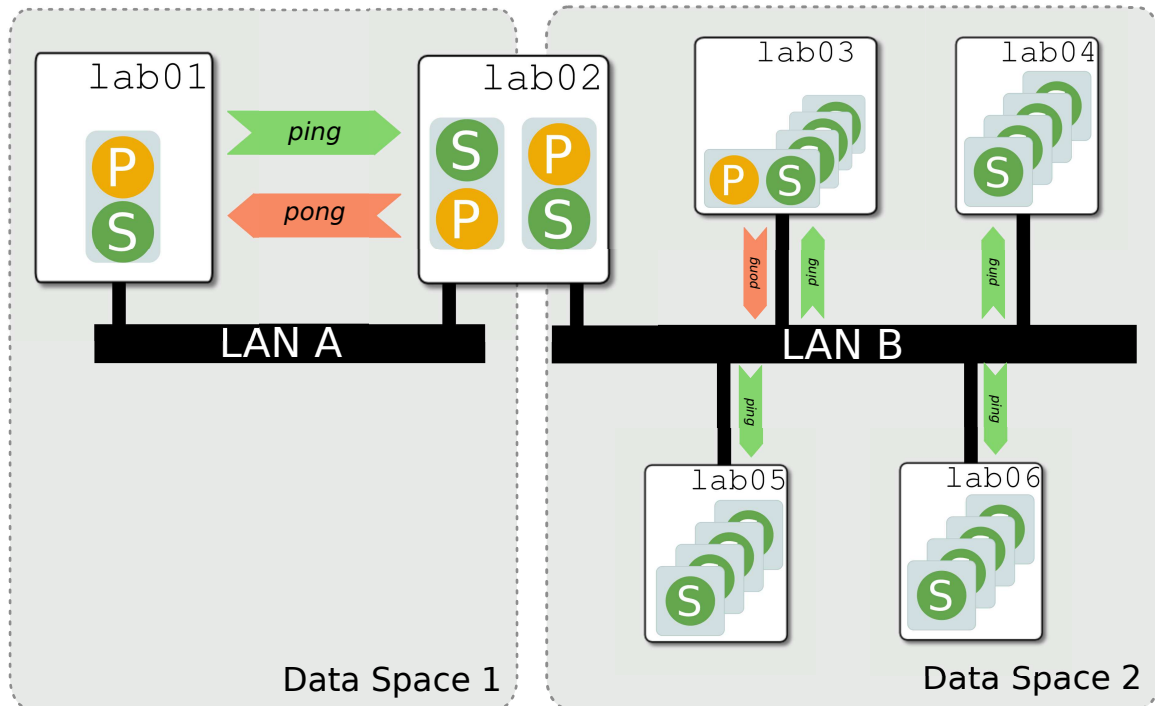


Figure 3.13: System topology used in 1 to N performance experiments.

3.7.3.1 DDS-IS performance impact

Figure 3.13 shows the topology used in this set of experiments. Specifically, we run a Ping topic publisher and a Pong topic subscriber on node lab01, a DDS-IS in node lab02 (for the no-DDS-IS scenario, IP forwarding was enabled at this node), and 0-4 subscribers in each of the nodes lab03, lab04, lab05, lab06.

To avoid overloading the DDS-IS with Pong samples, only one Pong publisher at node lab03 publishes Pong samples back to lab01 (for measuring average round-trip latency and throughput on lab01 node).

Figure 3.14 depicts average round-trip latency results obtained for this set of experiments. The obtained maximum overhead is 330 microseconds (for the 8192-bytes payload, 1 to 1 experiment). For the 4096-bytes, 1 to 4 case, the DDS-IS scenario obtains almost zero round-trip latency overhead (852 and 846 microseconds for the cases with and without DDS-IS respectively).

Indeed, for a number of subscribers greater than 4, and a payload greater than 4096 bytes, we have obtained lower round-trip latencies values for the DDS-IS scenario than for the no-DDS-IS one. This is because the latency overhead introduced by the DDS-IS logic is offset by the reduction of the load on lab01 (the original publisher), the reduction of traffic in LAN A (this will be studied in next section), and

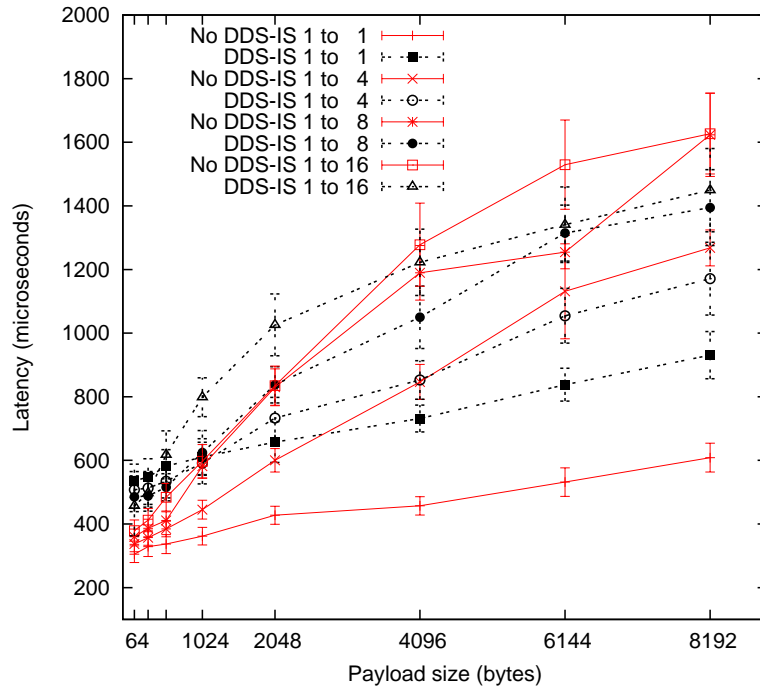


Figure 3.14: Average latency for different payloads when demultiplexing data from 1 to N.

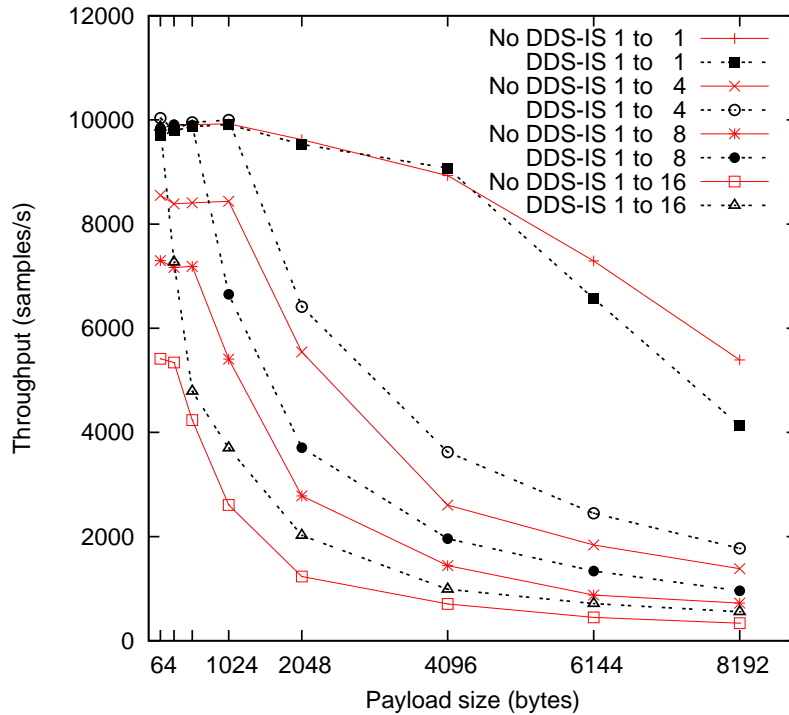


Figure 3.15: Throughput per subscriber for different payloads when demultiplexing data from 1 to N.

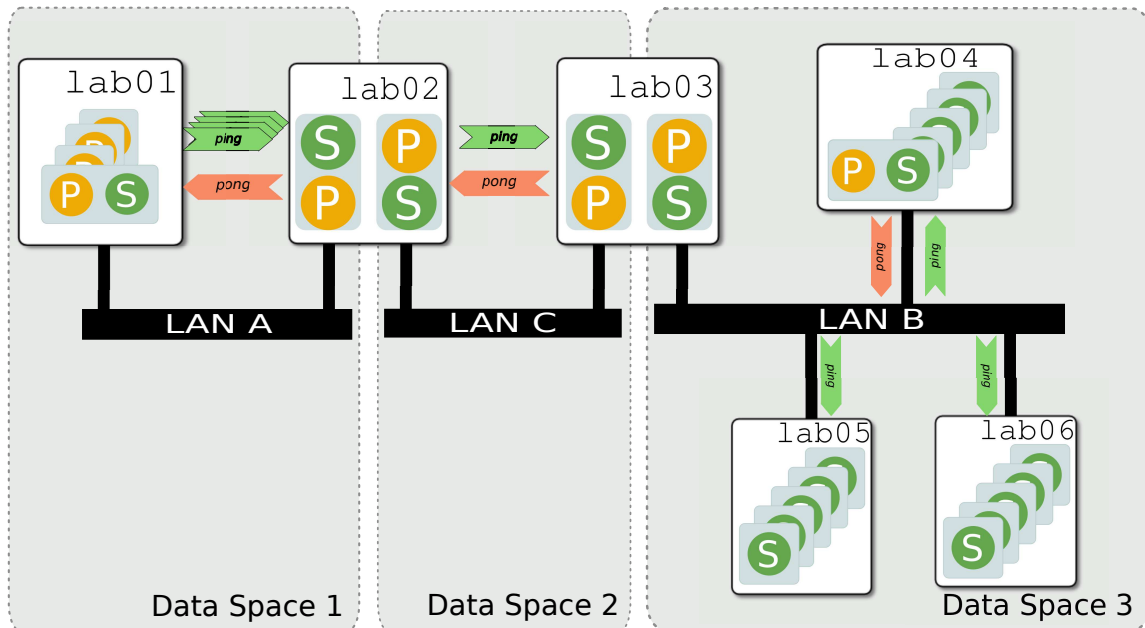


Figure 3.16: System topology used in M to N scalability experiments.

the fact that it is more efficient to re-send lost samples from lab02 (in the same LAN) than from lab01 (located at two hops).

Regarding throughput, the results in Figure 3.15 show that the DDS-IS scenario beats the no-DDS-IS scenario for all the experiments with a number of subscribers greater or equal than 4. This is especially relevant for the 64-byte, 1 to 16 scenario, where the DDS-IS maintains a rate of 10,000 samples per second per subscriber, while in the no-DDS-IS case the rate drops to 5400 samples per subscriber.

The explanation for this behavior is the same we stated for latency experiment: network usage reduction on LAN A, reduction of load for original source-node, and the fact that re-sending from lab02 is cheaper than re-sending from lab01.

3.7.3.2 DDS-IS scalability impact

Earlier in this chapter we stated that the DDS-IS can help in reducing the network traffic load.

The following experiment (Figure 3.16) supports our claim. The experimental set-up consists of 1-4 Ping publishers (running on lab01), 1-15 Ping subscribers (uniformly distributed among lab04, lab05, and lab06), and two DDS-IS (running on lab02 and lab03). To avoid overloading the DDS-IS with Pong samples, only one Pong publisher at node lab04 publishes Pong samples back to lab01.

Although many other alternative settings can be envisaged, this simple scenario is useful for our goal. The configuration for the no-DDS-IS scenario is almost the same, although lab02 and lab03 just forward the samples between LAN A and LAN B.

The scenario without DDS-IS exchanges all the samples through a unique DDS domain (this case corresponds to just one data-space with no interconnection service), whereas the DDS-IS scenario presents three different data-spaces: one between lab01 and lab02; another one between lab02 and lab03; and another one among lab03, lab04, lab05, and lab06.

This configuration allows checking the potential of the DDS-IS, its ability to multiplex (aggregate) and demultiplex (separate) DDS traffic. In this respect, with this experiment we evidence that DDS-IS eases the integration of DDS systems across Wide Area Networks (WANs).

Particularly, we point out that DDS-IS can isolate individual applications from WAN transport. For example, local applications can communicate using IP multicast for efficient data distribution while DDS-IS bridges remote sites using WAN infrastructures (maybe using reliable TCP connections).

The potential of the DDS-IS for reducing the network traffic can be explained using a simplistic example: assuming M publishers and N subscribers, a network without loss, and S number of samples published by each publisher, the number of total samples to the LAN C would be equal to $M \times N \times S$ in the scenario without DDS-IS. On the other hand, if we use the configuration shown in Figure 3.16, the traffic in LAN C decreases to $M \times S$, given that only one copy of each published sample is sent.

Additionally, according to the OMG standard DDS entities (publishers and subscribers) exchange discovery traffic. Whereas in the no-DDS-IS scenario this is a $M \times N$ problem, the DDS-IS only creates a DW on lab03 and a DR on lab02, what evidence that DDS-IS can reduce the complexity in LAN C to a much simpler discovery problem (1×1 for the particular setting considered).

Figure 3.17 shows the total generated DDS traffic load on LAN C for the scenarios with and without DDS-IS. As expected, the obtained results show the traffic in the no-DDS-IS scenario increases almost linearly with the number of subscribers, whereas the traffic in DDS-IS scenario is constant for a growing number of subscribers. Thus, finally we conclude that the DDS-IS can help in saving network resources.

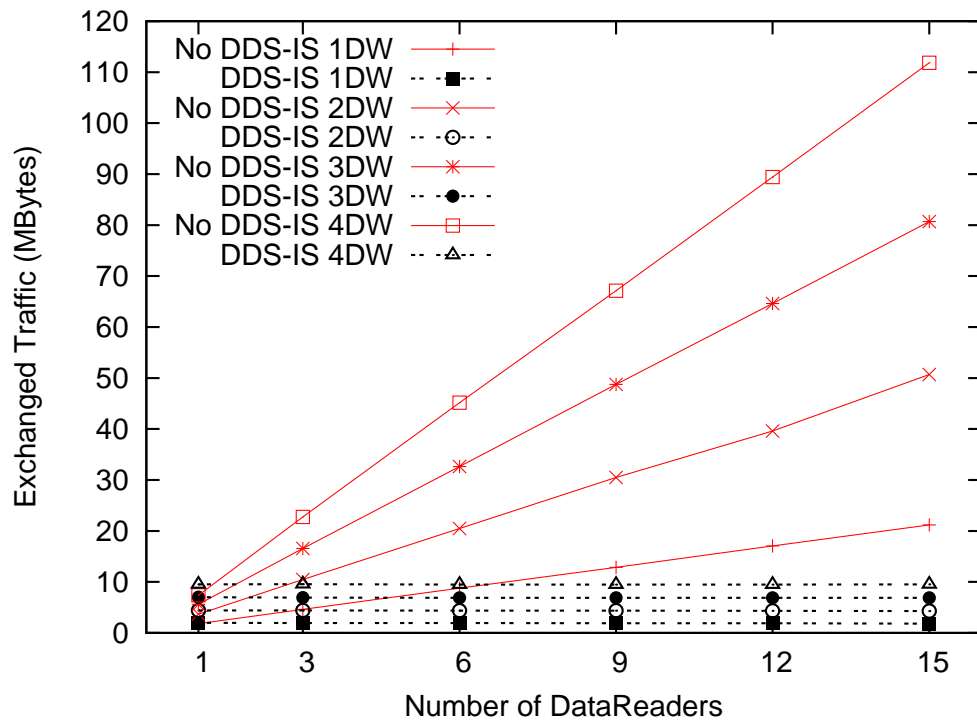


Figure 3.17: DDS traffic load on LAN C for both no-DDS-IS and DDS-IS scenarios.

3.8 Impact on DDS QoS policies

To finish this section, we now study the impact of the DDS-IS on DDS QoS policy enforcement.

Table 3.8 studies the level of enforcement for each QoS policy: local (if a particular QoS is enforced locally by DDS middleware), hop-by-hop (if a particular QoS is enforced within a particular DDS domain), or end-to-end (if a particular QoS is enforced across interconnected domains). The table also contains a description of each QoS policy, and its corresponding compatibility checks.

In general, QoS enforcement is applicable only among entities within a particular domain (*i.e.*, hop-by-hop) for most of DDS QoS policies. There are, however, two exceptions: LIVENESS and LIFESPAN. These two DDS QoS policies can be enforced on an end-to-end basis (*i.e.*, across multiple interconnected domains).

Table 3.1: Study of the level impact of DDS-IS on DDS QoS policies.

Name	Description	Compatibility requirements	Level of enforcement
DEADLINE	Maximum time between two updates	Offered <code>deadline_period</code> \leq requested <code>deadline_period</code>	Local According to the standard, each individual publisher and subscriber enforces this QoS. In this sense, if DEADLINE period is missed, the particular publisher or subscriber generates a notification to its associated application.
DESTINATION_ORDER	Controls how each subscriber resolves the final value of a data instance that is written by multiple publishers.	Offered kind \geq requested kind (BY_SOURCE_TIMESTAMP $>$ BY_RECEPTION_TIMESTAMP)	Local Each individual subscriber will ensure samples are delivered to application according to this QoS.
DURABILITY	Controls whether DDS will make data available to late-joining readers. If this QoS is set to PERSISTENT, publishers send a set of past values (the actual number of samples depends on HISTORY QoS) to late joining subscribers.	Offered kind \geq requested kind (PERSISTENT $>$ TRANSIENT $>$ TRANSIENT_LOCAL $>$ VOLATILE)	Hop-by-hop Historic samples are sent between publishers and subscribers within the same domain. Although samples published within a domain will be normally bridged by the DDS-IS, there is not mechanisms for ensuring that those samples arrive to their final destination (see RELIABILITY QoS), even if both domains have DURABILITY set to PERSISTENT.
DURABILITY_SERVICE	Configures both HISTORY and RESOURCE_LIMITS	None	N/A
			Continuing next page

continued from previous page			
Name	Description	Compatibility requirements	Level of enforcement
ENTITY_FACTORY	Controls the behavior of a DDS Entity as a factory for other DDS entities.	None	N/A
GROUP_DATA	Attaches additional information to the created publishers and subscribers.	None	N/A
HISTORY	Controls the number of historic samples stored by publishers.	None	N/A Each individual publisher will store the a number of historic samples equal to the value of this policy.
LATENCY_BUDGET	Provides a hint about the priority of data-communications	Offered duration \leq requested duration	N/A (hint) According to the standard, this policy is considered a hint. There is no specified mechanism as to how the service should take advantage of this hint.
LIFESPAN	Limits the validity of each individual update (comparing samples timestamp with reception date).	None	Hop-by-hop/End-to-end The level of enforcement of this QoS depends on the configuration of the <code>publish_with_original_timestamp</code> DDS-IS parameter. <code>publish_with_original_timestamp</code> controls whether samples maintain the original timestamp or if the timestamp is regenerated with each hop.
			Continuing next page

continued from previous page			
Name	Description	Compatibility requirements	Level of enforcement
LIVELINESS	Controls the mechanism and parameters used by the Service to ensure that particular entities on the network are still alive.	Offered kind \geq requested kind (MANUAL_BY_TOPIC > MANUAL_BY_PARTICIPANT > AUTOMATIC) Offered lease-duration \leq requested lease-duration	Hop-by-hop/End-to-end DDS-IS supports the propagation of DDS entities creation and disposal across the bridged domains. In addition, the DDS-IS could be configured for maintaining DDS-IS publishers and subscribers independently of the liveliness state of the entities within bridged domains.
OWNERSHIP	Controls whether DDS allows multiple publishers to update the same instance	Offered ownership $==$ requested ownership (SHARED or EXCLUSIVE)	Hop-by-hop
OWNERSHIP_STRENGTH	Determines the ownership of a data-instance.	None	Hop-by-hop Each individual subscriber delivers to the application only the samples generated by the publisher with a higher OWNERSHIP_STRENGTH.
PARTITION	Subdivides a domain into subdomains.	Publishers and subscribers must use the same partition for communicating.	Hop-to-Hop DDS-IS entities and DDS application entities only communicate if they are associated to the same PARTITION.
			Continuing next page

continued from previous page			
Name	Description	Compatibility requirements	Level of enforcement
PRESENTATION	Defines the dependence across updates	Offered access.-scope \geq requested access.-scope (GROUP $>$ TOPIC $>$ INSTANCE)	Local Each individual subscriber ensures that samples are delivered to the corresponding application according to this QoS.
READER_-DATA_LIFECYCLE	Controls the behavior of the subscriber with regards to the lifecycle of the data-instances it manages	None	N/A
RELIABILITY	Ensures published samples are delivered to subscribers.	Offered kind \geq requested kind (RELIABLE $>$ BEST_EFFORT)	Hop-by-hop Although DDS will ensure that samples published in a particular domain arrive to all the subscribers within that domain, there is not mechanisms for ensuring end-to-end enforcement of this policy. In this sense, if a DDS-IS along the datapath crashes, the original publisher has not means for checking that samples have been received across bridged domains. Fortunately, this problem can be mitigated using redundant DDS-IS nodes.
			Continuing next page

continued from previous page			
Name	Description	Compatibility requirements	Level of enforcement
RESOURCE_-LIMITS	Controls the resources consumed by DDS.	None	N/A Each individual publisher or subscriber will limit the maximum allocated resources according to this policy.
TIME_BASED_-FILTER	Indicates the minimum separation between two updates delivered by DDS to the application.	deadline period \geq minimum_separation	Local Each individual subscriber ensures that samples are delivered to application according to this QoS.
TOPIC_DATA	Attaches additional information to the created Topics.	None	N/A
TRANSPORT_-PRIORITY	Allows the application to take advantage of transports capable of sending messages with different priorities.	None	N/A (hint) According to the standard, this policy is considered a hint. There is no specified mechanism as to how the service should take advantage of this hint other than perhaps propagate it.
USER_DATA	Attaches additional information to the created Entities.	None	N/A
WRITER_-DATA_LIFECYCLE	Controls the behavior of the publisher with regards to the lifecycle of the data-instances it manages	None	N/A

* For the sake of simplicity, we have assumed that publisher = DataWriter and subscriber =

DataReader.

End-to-end QoS enforcement for all of the DDS QoS policies is out of the scope of DDS-IS, as it would require modifying the behavior of the DDS middleware itself. Besides, it would break backwards compatibility, which is stated in requirement R6. However, in the next chapter we propose a protocol for providing end-to-end signaling, and which allows DDS entities to enforce QoS policies at end-to-end level. For an analysis of the impact of DDS-IS on each DDS QoS policy please refer to Section 3.8.

3.9 Conclusions

In this chapter we propose the DDS-IS, a solution to address one of the key limitations in DDS deployments: the interconnection of DDS data-spaces.

In addition to the interconnection problem, the DDS-IS also addresses three unresolved issues of DDS: data transformation between data-spaces, system scalability, and QoS adaptation between remote entities. Moreover, following the data-centric philosophy of DDS, the proposed service is able perform content-based filtering of the exchanged data.

The design of the DDS-IS features an efficient data transforming capability, which transforms data as it flows between applications. As a result, multiple versions of the same application, or even applications of different vendors, can now interoperate, without change, despite differences in data structures and interface definitions.

DDS-IS also supports content-based filtering, which can be used for avoiding sensitive information of a given data-space to be exposed to other data-spaces.

Regarding DDS QoS profiles, the DDS-IS relies on DDS QoS compatibility checks, enforcing the compatibility among bridged applications. In order to keep QoS consistent across bridged data-spaces, the DDS-IS does not connect DDS applications whose QoS are not fully compatible with the ones configured for the DDS-IS.

One of the key benefits of our solution is its standards compliance. The DDS-IS is fully compliant with the latest specifications of the DDS standard family. As a result, our design is not implementation-dependent, and hence it is applicable to any existing or future DDS implementation. Namely, the DDS-IS benefits from the DDS-XTypes, a new OMG specification that is in the latest stages of its acceptance, for accessing to topic types discovered at execution time.

We have implemented a DDS-IS prototype that demonstrates the advantages of our proposal. This prototype is compliant with the DDS-RTPS [69]. Based on this prototype, we report experimental results to evaluate the performance impact of the

proposed service. The reported results show that the DDS-IS has a very low impact on DDS performance, in terms of overhead latency and achievable throughput.

We have also demonstrated that the DDS-IS can improve the scalability of DDS systems by reducing the network traffic load between remotely interconnected data-spaces.

Despite of the novel features of DDS-IS, several DDS issues remain unsolved. In particular, end-to-end entities and data-content signaling and QoS enforcement are not solved yet. In addition, DDS systems does not scale well for big scale deployments. In the following chapters we propose solutions to those identified problems.

Data-Centric SIP (DCSIP)

AFTER the design, implementation and evaluation of the DDS-IS in Chapter 3, in the following two chapters we address relevant DDS issues that deserve further consideration.

In particular, in this chapter we propose a universal naming scheme for DCPS entities and resources. In addition, we establish the basis of a protocol for session signaling and resource discovery in medium-scale DCPS environments.

In the next chapter we focus in solving a still open issue in DCPS environments: their scalability in big-scale deployments.

4.1 Motivation

Originally, the DDS specification standardized the programming model and API for developing DCPS applications. Despite of the fact that it defined the entities for exchanging discovery information, it did not specify the network protocol used by DDS implementations in a way that would allow different implementations to interoperate. The OMG addressed this issue by specifying the DDS-RTPS [69].

The DDS-RTPS specification defines the so called DDS SDP to perform the discovery. This discovery protocol was designed for relatively small environments (up to a few thousands nodes connected through a LAN). In this regard, the current DDS specifications do not address the problem of universal entity identification and universal data-content identification.

Another existing limitation impacts scalability. Most of current DDS implementations rely on using multicast or a centralized server for discovery. In this sense, DDS-RTPS does not address the problem of entity or data-content discovery in big deployments, nor provides any mechanism for performing NAT traversal. Fortunately, the DDS-RTPS specification allows DDS implementations to define their own discovery protocols, therefore is possible to define new protocols that address these issues.

The introduction of the DDS-IS creates new problems. As we studied in Chapter 3, current DDS QoS policies were designed to work in LAN-based scenarios, and they do not work correctly in environments where a DDS-IS node is present.

Another problem is the control of data propagation across different data-spaces. The DDS-IS provides mechanisms for controlling what data is bridged between two data-spaces. However, systems administrators have to configure DDS-IS nodes manually, as there are no standard mechanisms that allow applications to negotiate DDS-IS routes.

In short, the introduction of DDS-IS nodes creates the need of new mechanisms that support the negotiation of QoS policies among remote nodes, along with the creation of publications and subscriptions across DDS-IS nodes. In order to address these issues, this chapter defines and proposes the use of a signaling protocol for DDS.

The remainder of the chapter is organized as follows. In Section 4.2 we present a motivating use case. In Section 4.3 we identify the limitations of DDS and DDS-IS we need to overcome. In Section 4.4 we discuss multiple considerations and adopted design decisions. In Section 4.5 we study the fundamentals of the proposed protocol, and in Section 4.6 we define the format for addressing content and DCPS entities. In Section 4.7 we define the different roles nodes can adopt in our proposal, and in Section 4.9 we study the system operation. In Section 4.10 we compare our proposal and SIP, and in Section 4.11 we compare our proposal and current DDS specifications. Finally, in Section 4.12 we summarize the main conclusions of the chapter.

4.2 Motivating use case example

In this section we provide a motivating use case scenario that helps to understand the functionality and advantages of our solution.

The chosen use case scenario is the Spanish Public Health Service (SPHS). Specifically, we will focus on the medical record databases, which store known allergies and past health problems of each patient.

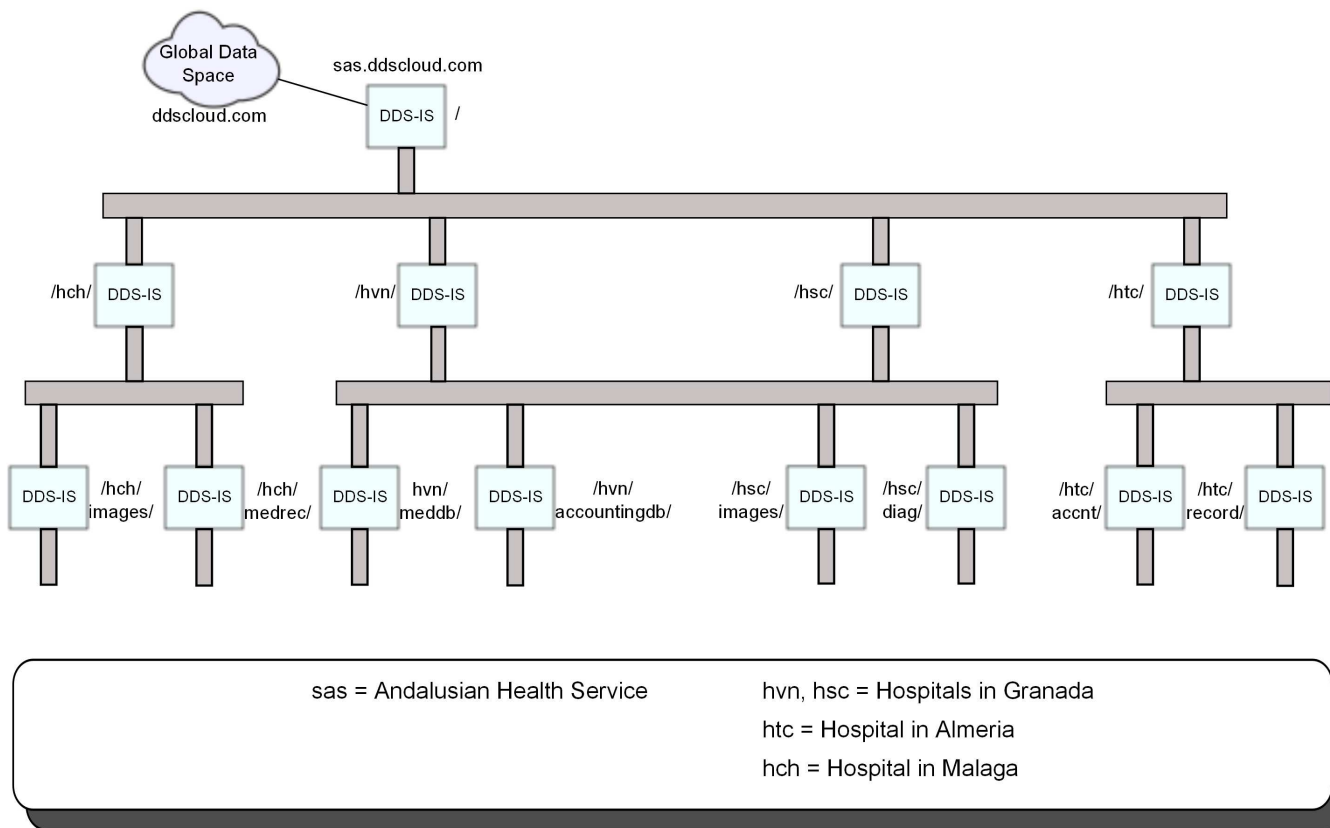


Figure 4.1: Topology of the Health Service use-case.

The current SPHS is divided in districts (each one managed by a regional government). The problem that arises in this scenario is the lack of an unified medical record database. Consequently, when a patient changes his place of residence he has to start a new medical record (sometimes even repeating expensive medical tests). Even worse, medical records of different hospitals belonging to the same district are not unified either.

Although it is clearly an inefficient system, its perpetuation is mainly due to two reasons: first, hospital administrators do not want to lose the control of their databases; and second and most important, each hospital uses its own database, thus making them incompatible.

We claim that data-centric approaches, and particularly DDS and DDS tools, can help to solve the interoperability problem while maintaining the autonomy of each hospital. This will enable the exchange of medical information between hospitals, saving the cost of unnecessary redundant medical tests.

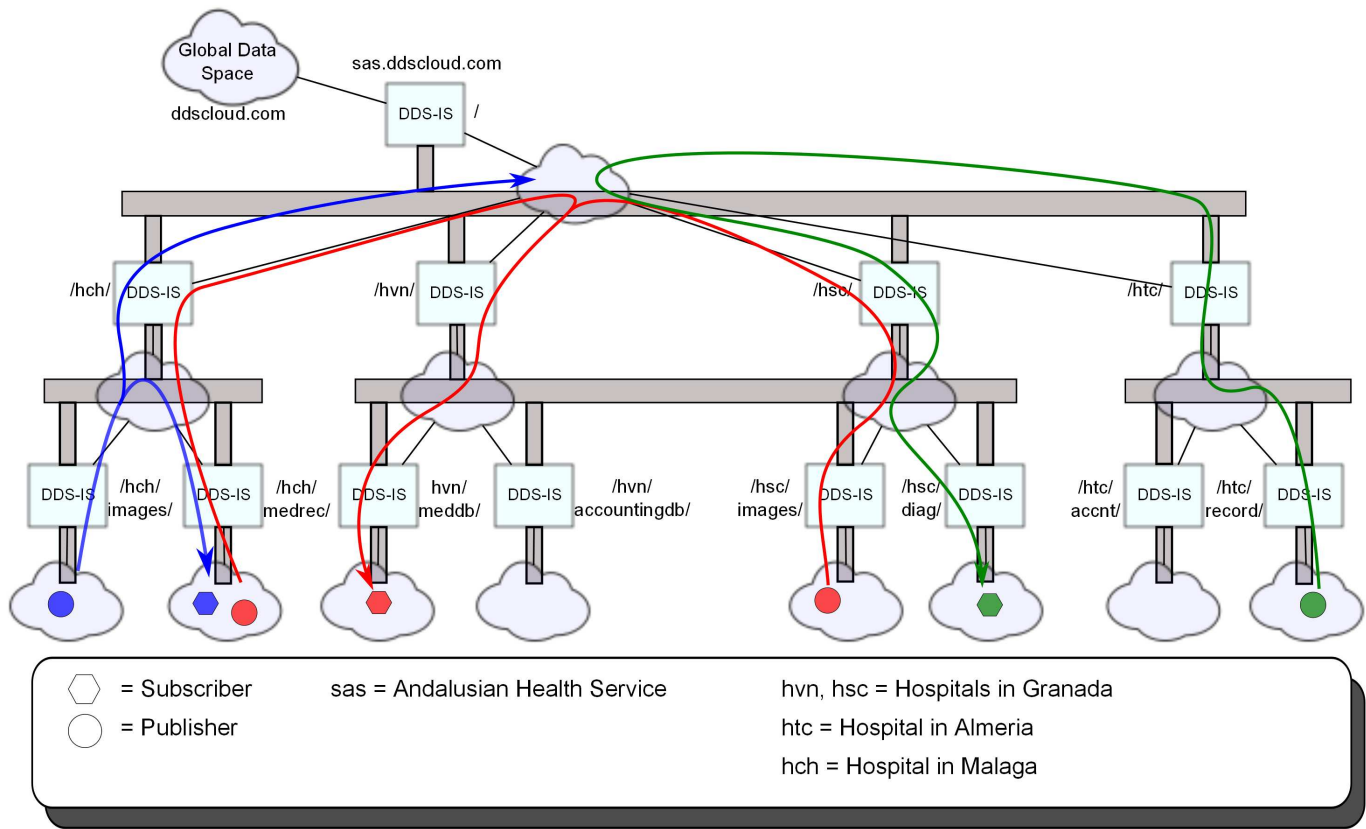


Figure 4.3: Current solutions provoke unnecessary and redundant traffic in the higher levels of the DDS-IS hierarchy.

by defining generic route creation rules, this solution is not scalable and does not adapt well to system evolution (*i.e.*, if system requirements change, the administrator will have to modify the existing rules in each affected DDS-IS to meet the new set of requirements).

Figure 4.3 depicts an example of the discussed limitations. In this example, system administrator has configured DDS-IS for creating upward routes upon simple discovery match (*i.e.*, after discovering a DDS entity in one of the bridged data-spaces), while downward routes are created only after a double match (*i.e.*, after discovering compatible DDS entities in the two bridged data-spaces). This avoids configuring routes manually, while avoiding to flood all the data-spaces with unnecessary samples.

We can see in the example scenario that the hch hospital *images-publisher* generates data-content (represented as a blue arrow) consumed by a *medrec-subscriber* in the same hospital. We can see that –as a result of the permissive DDS-IS configura-

tion— this data-content is adapted and forwarded to the highest-level of our topology (named `sas.ddscloud.com/`). We can see also that the red arrow does not follow the optimal way from `hsc` to `hvn`, as it also traverses the data-space associated to node `sas.ddscloud.com/` (instead of using the LAN segment these hospitals share). Regarding the green arrow, it represents a topic that is being forwarded between two data-spaces, even if the subscriber does not currently need more information than the generated in its local data-space.

In order to solve these DDS limitations, in this chapter we propose DCSIP, a protocol for resource discovery and session control in DCPS systems.

Figure 4.4 shows how using DCSIP improves DDS scalability. In particular, DCSIP avoids flooding data-spaces with unrequested samples (in our example, the blue and green arrows do not reach the data-space associated to node `sas.ddscloud.com/`). In addition, our solution allows for sending data-content samples over optimized routes, avoiding to overload unconditionally the nodes located in a higher hierarchy level. In our example, the system directly bridges samples from `sas.ddscloud.com/hsc/images/` to `sas.ddscloud.com/hvn/meddb` without traversing `sas.ddscloud.com/hsh/`.

The proposed DCSIP-based solution solves the studied problems, as it provides a means for negotiating a transient DDS-RTPS flow between publishers and subscribers located in remote data-spaces. Local DDS entities can continue using regular DDS discovery for exchanging data-content. In the same way, DDS entities that communicate through very static and immutable routes and do not need end-to-end QoS enforcement can keep using regular DDS discovery. On the other hand, applications that need to create dynamic routes for establishing temporal DCPS/DDS-RTPS sessions with end-to-end QoS constraints (for example, for transferring a backup of a data-base, a set of values upon request, or a multimedia session over DDS) can greatly benefit of a signaling protocol for DDS.

4.3 New required features

4.3.1 Limitations of DDS and DDS-IS

The DDS-IS proposed in the previous chapter provides the service for bridging different DDS domains and topics. However, the following problems remain unsolved:

1. **Universal topic and domain identifiers:** Neither the DDS specification nor any existing DDS implementation defines a universal resource locator for naming DDS entities and data-content. DDS deployments have been tradition-

developed by using static topics and data-types. Although the definition of DDS-XTypes (see Section 2.2.4) allows applications to register topics and data-types on run-time, there are no mechanisms that allow applications for dynamically negotiating topics, data-types, and QoS policies.

4. **End-to-end DDS QoS enforcement:** DDS QoS approach is more focused on reactive mechanisms (such as dropping expired samples or retransmitting lost samples) than on proactive mechanisms (resource provisioning). In fact, the only proactive mechanism provided by the service is QoS compatibility checking (*i.e.*, DDS middleware just checks if the QoS associated to two communicating DDS entities are compatible, not establishing the communication if they are not fully compatible). In this regard, there is not a mechanism for QoS negotiation beyond the DDS QoS compatibility checks. Another related problem is the lack of end-to-end QoS support (given that QoS is enforced in a hop-by-hop basis).

4.3.2 Solution requirements

In order to solve the stated problems we propose DCSIP, a signaling protocol designed for working over DDS systems. DCSIP must fulfill the following requirements:

- R1: Interdomain entities discovery.** The proposed protocol must provide mechanisms for announcing DDS publishers and subscribers among different DDS domains. This feature will allow DDS entities located in different (and possibly remote) DDS domains to discover each other. The announcement samples should traverse DDS-IS.
- R2: DCPS session signaling.** The proposed protocol must provide session signaling for DDS systems. A DCPS session is an abstraction that defines a common state between multiple applications/participants that share one or more topics. The negotiation of a DCPS session between two remote nodes will provide intermediate DDS-IS nodes the necessary information for creating topic routes according to application requirements.
- R3: Topic negotiation.** The proposed solution must provide mechanisms for negotiating the topic names and types associated to a DCPS session. In particular, a DCPS session must allow applications to control the creation and destruction of a group of publish-subscribe associations among multiple participants. This will allow developers to deploy multimedia applications over DDS by using mechanisms similar to the ones used in SIP systems.

R4: End-to-end QoS negotiation and enforcement. Our signaling protocol will be able to negotiate QoS between remote DDS applications, and to enforce QoS in an end-to-end basis.

R5: DDS-based design. Our solution will be completely based on DDS standard family. This will allow our solution to benefit from existing DDS/DDS-IS features (content-based filtering, data-content history, data transformations). Moreover, this will ease its integration in existing deployments, as it will work without modifying existing NAT and firewall configurations.

4.4 Design decisions

To solve the evidenced issues in the use case introduced in the previous section, we present a solution for medium-scale DCPS deployments that covers information management, signaling, and routing. For each one of these aspects, in this section we discuss multiple considerations and adopted design decisions.

4.4.1 Information management

Prior to specifying a signaling protocol, we need to define how to organize the information. In particular, we adopted the following design decisions:

- **Hierarchy:** Our system is organized into a tree-like structure, being the realm root the highest hierarchical level. At the realm root, a rendezvous server manages the registration of DCPS entities and resources. Each node (apart from the root) has at least one parent and any number of children.
- **Named-based:** System administrator assigns a local identifier (a label) to each node. This label is announced both to node's parents and to node's children in order to build a unique label chain. The use of labels is compatible with the *Named Data Networking (NDN)* [109], and will allow to use DDS over *NDN* networks in the future.
- **Autoconfiguration:** Using labels, nodes dynamically build a unique name that identifies the location of each node in the system (the label chain). A label chain describes the logical path from the realm root to a specific node, and it is updated each time a node changes its location.
- **Mobility:** Since the label chain is built dynamically, our information management model supports mobility. In this way, if a node identified by label `meddb` and label chain `sas.ddsccloud.com/hvn/meddb` moves to the data-space

associated to hch node, the label chain for meddb will dynamically change to `sas.ddscloud.com/hch/meddb`. The only requirement is that the label is not already in use in the same data-space by another node.

- **Scalability:** Our solution increases DDS scalability, as it allows applications to limit the number of data-samples that are flooded to the network. This is because the system does not propagate DDS discovery and data-content information until two endpoints have declared their intention to share data-content. However, this proposal relies on a centralized server for storing rendezvous information. In this regard, in Chapter 5 we will propose a complementary P2P-based solution for big-scale deployments.

4.4.2 Signaling

Once we have defined how the information is structured, we must define how to control the information exchange. To fulfill this goal, in the following sections we define DCSIP, a signaling protocol for announcing data-content, data consumers, and data producers; and for establishing DCPS sessions.

We have considered two approaches for solving signaling propagation, both supported by our solution:

- **Using a dedicated logical topology for signaling:** This approach isolates the propagation of signaling and data-content by using a specific DDS domain for signaling information. This approach is similar to the one adopted in SIP, where the end-to-end media packets (usually transmitted over Real-time Transport Protocol (RTP)) take a different path from the SIP signaling messages.

Since the signaling domain is well-known *a priori*, DDS-IS nodes can start exchanging signaling information without any additional configuration. In fact, nodes can exchange signaling information even with nodes not associated to the same data-content domain. This allows the system to keep existing data-content flows isolated, even using different port ranges for data-content and signaling, while administrators are still able to create optimized routes, if necessary.

In addition, as every single node is connected through a global signaling data-space, it is possible to calculate the optimal route at signaling level, and then to open the proper routes for data exchange. However, these routes may require the creation of new participants when the involved DDS-IS nodes do not share any existing domain, and this is expensive.

- **Using the same logical topology for both signaling and data-content routes:** This approach eases the integration of DCSIP with existing deployments, as it does not require creating new dedicated signaling domains. Moreover, it consumes less resources, as it does not require creating the additional pair of participants needed to handle signaling domains. It also prevents DDS-IS nodes sharing the same physical network from communicating without an explicit permission.

However, the level of isolation between data-content and signaling is lower than in the previous case. In addition, and since there is no global signaling data-space, the achievable level of route optimization is limited by the number of alternative connections that the system administrator defines for each node (*e.g.*, an administrator can configure two DDS-IS nodes for using independent domains by default while they use a common domain for establishing optimized routes).

4.4.3 Routing

Once two remote entities decide to communicate, there are multiple alternatives for establishing a route, all supported by the proposed DDS-IS (described in Chapter 3). These alternatives have to be supported by DCSIP:

- **Using dedicated domains:** This is the most expensive alternative, as it requires to create a new dedicated participant for each new route a DDS-IS creates. This approach does not allow DDS-IS nodes to send multiple flows using the same participant (*i.e.*, using the same DDS domain), since flows would not be distinguishable after their aggregation.

However, this approach has also advantages. It eases the isolation between DDS entities (participants, DWs and DRs) that have access to different data-content. Since it uses an independent domain for each route, neither the data nor the discovery information will be accessible to entities not included in the route. In addition, since sessions are isolated at domain level, two applications can share multiple topics within the same session by just configuring the same domain for all the involved topics.

- **Using dedicated topics:** This approach uses a dedicated topic for each route. This approach is less expensive than using domains, as it does not require creating additional participants. However, the level of isolation is lower than in the previous case because entities that do not have any common route can still exchange discovery messages at participant level. Since this approach uses topics for identifying routes, the topic name for intermediate hops is associated to the session and is different to the original topic name. Therefore, it

is necessary that the first and last DDS-IS perform a conversion to the original topic name. This approach does not allow DDS-IS nodes to send multiple flows using the same DW/DR (*i.e.*, using the same topic), since flows would not be distinguishable after their aggregation. However, routes are multiplexable at participant level, and multiple flows can share the same data-spaces within certain sections of the route-path.

- **Using content-filters and topic keys:** This uses a combination of content-filters and topic-keys for routing the data-content. The level of isolation of this approach is similar to the one using topics, as entities that do not have any common route can exchange discovery messages at participant level. Since this approach relies neither on domains nor on topics for flows identification, it is not necessary to perform any conversion to the exchanged data. Although this approach increases the CPU load in DDS-IS nodes due to content-filter evaluation costs, it decreases the network traffic. This approach allows for multiplexing flows without mixing them up. This enables to aggregate multiple flows that share the same routing path, therefore reducing the overall discovery and data-content traffic.

In the following sections we focus on the information management and signaling aspects of our solution. Regarding routing, it is mostly based on the existing DDS-IS, and it only requires to add DCSIP support to existing DDS-IS functionality.

4.5 DCSIP fundamentals

DCSIP takes multiple ideas from the SIP protocol [83], hence, it eases the integration of DCSIP-based systems with SIP-based environments, or even with future P2P-SIP based deployments.

Despite of the similarities between DCSIP and SIP, there is one major difference: while SIP has been designed to operate within the request-response paradigm, DCSIP follows a DCPS approach. As described in Section 2.2.1, the DCPS approach provides time, space, and synchronization decoupling as well as the advantage of that data are not opaque to the DDS middleware. Below, we further explain the implications of the DCPS approach.

In order to understand the proposed protocol, in the following sections we define the set of basic concepts around DCSIP has been designed.

4.5.1 Realm

A **realm** is a set of nodes with a common Domain Name System (DNS) namespace and rendezvous server. The highest level of the realm hierarchy is the realm root, identified by a DNS domain name followed by the '/' character (e.g., `sas.ddscloud.com/`).

4.5.2 Scope, scope-label, and scope-path

In SIP, resource locations are defined by DNS names or IP addresses. Our solution, as we stated in see Section 4.4.1, specifies location by using a concatenation of labels. In this section we specify how we use those labels to define a particular location or a route between two locations.

Scope-label is a string that identifies a node within a DDS data-space (e.g. `hvn`). System administrators must set the scope-label during the system initial deployment.

Scope-path is a concatenation of scope-labels (each one associated to a node), and it represents a route between two nodes. Scope-path has two objectives. First, it determines how to route DCSIP messages; second, it limits the propagation of both signaling messages and data-content samples, avoiding the need of flooding the network with unnecessary messages.

Scope is an identifier that univocally identifies a node within a realm, therefore allowing other nodes to send samples to that particular node. A scope is a scope-path representing the sequence of nodes located between the root of the realm and the node identified by the scope.

4.5.3 Full Qualified Data-space and Topic Names

In SIP, users and UAs are identified using a DNS name or an IP address. However, this approach is not valid for a DCPS system (which is focused on data-content instead of data-location). In this section we propose a unique identifier for DCPS domains and topics, and in the next section we focus on DCPS entities identification.

As we stated in 4.3.1, DDS lacks of a universal identifier for topics and domains. To solve this, we propose the concept of **Full Qualified Domain Name (FQDN)**. FQDN is an identifier that uniquely identifies a particular data-space. It consists of two parts. First, a DNS namespace associated to a particular realm; second, a data-space name. We study the specific format of the FQDN later in Section 4.6.

Full Qualified Topic Name (FQTN) is an identifier that univocally identifies a data topic. Since a topic is a segment of a data-space, we decided that each FQTN must explicitly reference the FQDN the topic belongs to. We study the specific format of the FQDN later in Section 4.6.

4.5.4 DCPS Entity Group and DCPS Entity Name

We can group DCPS entities using two characteristics: their kind (*i.e.*, participant, publisher, or subscriber), and the data-segment they access (*i.e.*, a data-space or a particular topic). This classification is relevant for our system, since entities of the same kind and data-segment will be usually interested in discovering the same set of DCPS entities.

In order to identify groups of nodes with the same interests we have defined the **DCPS-Entity-Group**. It is an identifier shared among a group nodes that host a particular group of DCPS entities. Specifically, each group contains either a set of publishers of the same topic, a set of subscribers of the same topic, or a set of participants of the same data-space.

DDS-RTPS defines the GUID, a sequence of 16 bytes, for univocally identifying DDS entities within a data-space. DDS-RTPS GUID is not human readable, and it can not include descriptive information for an entity. Consequently, GUID is not suited to create lists of sensors and actuators that are readable and meaningful for human administrators.

To overcome the limitations of DDS-RTPS GUID, we propose the concept of **DCPS-Entity-Name** for identifying DCPS entities. As we will study in Section 4.6, DCPS-Entity-Name is a URI that univocally identifies a participant, publisher, or subscriber within a scope. Consequently, it can include a meaningful name that helps human administrators to identify the associated entity (*e.g.*, `../temperature-sensor01`).

4.5.5 DCPS session

DCPS session defines a common state between two nodes. This common state is associated to a set of QoS policies and other DDS parameters (topics, data-types). It is an abstraction that defines a common state between multiple DCPS entities that share data-content. A DCPS session allows applications to manage the creation and destruction of groups of publish-subscribe exchanges among multiple DCPS entities.

A DCPS session is similar to a SIP session. However, DCPS sessions are adapted

to the unique characteristics of DCPS environments. In this way, a DCPS session is associated to a set of FQDN/FQTN and data types rather than the IP addresses, ports, and codecs associated with a typical SIP session. In the same manner, a DCPS session is bound to a set of publications, subscriptions, and possibly DCPS routes between two participants, whereas a SIP session is generally associated to a set of RTP streams exchanged between two peers.

4.6 Resource naming format

In our proposal, we use an URI-based format for identifying DCPS resources. This decision is due to the following reasons:

- The format is flexible enough to support scenarios with different hierarchy levels (from scenarios where only one data-space is associated to the realm, to scenarios where there are several sub-levels of data-space).
- This format is easily readable and intuitive.
- The proposed URI format is easy to map into CoAP URI format [97], which is an IETF solution for accessing to constrained devices' (*i.e.*, sensors and actuators) information.

In our proposal, the URI for a FQDN is built as follows:

```
dcps:// < realm > / < data-space-name > /
```

Where `dcps` stands for "Data-Centric Publish-Subscribe" and `data-space-name` could be a hierarchy of data-space names (separated by forward slashes), for example:

```
dcps://ugr.es/labs/microwaves/
```

Since a topic is a segment of a data-space, the URI for a FQTN is built by attaching the topic-name to the FQDN of the topic's data-space:

```
dcps:// < realm > / < data-space-name > / < topic-name > /
```

DCPS-Entity-Group URIs are built by adding `/.pub/`, `/.sub/`, or `/.par/` suffixes to a FQDN or FQTN URI. As an example, the URI for accessing to the information published to temperature topic from the microwaves lab within the University of Granada, could be:

```
dcps://ugr.es/labs/microwaves/temperature/.pub/
```

This unique URI address refers to all the discoverable publishers that send updates to the temperature topic within the `ugr.es/labs/microwaves` data-space.

Our design also uses this URI format for the DCPS-Entity-Names (see Section 4.5.4). In order to save storage space and traffic load, our system stores DCPS-Entity-Names as relative URIs (using the **scope** URI as base). For example, if the following DCPS-Entity-Name

```
./temperature-sensor01
```

is associated to the scope with URI `dcps://ugr.es/labs/wmnlab/`, the absolute URI for the entity will be:

```
dcps://ugr.es/labs/wmnlab/temperature-sensor01
```

4.7 Node roles

Prior to explain message functionality and format, we identify the different roles that nodes can assume within a DCSIP dialog:

- **Endpoint Nodes:** Nodes that run DCPS applications and that do not necessary perform system maintenance operations. We define two types of endpoint nodes:
 - **I-Node:** This is the initiator of a DCSIP request. It usually requests for the establishment of a DCPS session, or the location of a particular resource.
 - **T-Node:** This is the target of a DCSIP request.
- **Network Nodes:** Nodes that perform system maintenance operations, and that route data-content. We define two types of network nodes:
 - **R-Node:** This is an intermediate node that processes DCSIP messages and routes those messages according to their content. This role is usually performed by a DDS-IS node.

- **Rvz-Node**: This is the rendezvous server of the realm. It stores a dictionary table containing pairs of FQDN/FQTN and a list of scopes.

4.8 DCSIP messages

In this section we define the set of messages that DCSIP nodes can exchange. We describe the format of DCSIP request and response messages.

Responses are preceded by “R_“, and are routed using their `forwarding_scope_path` field. In a response, the I-Node and T-Node are the same than in the original request.

In following subsections we provide a brief explanation for each DCSIP method and we list its associated fields. However, for the sake of clearness, we do not include the definition of the fields. For more information about DCSIP field definitions, please refer to Table 4.1 in Section 4.10.

4.8.1 Joining and registration

In this section we describe the DCSIP methods for node joining and content registration.

- **SCOPE_ANNOUNCEMENT**: This method announces a node’s scope-label and scope to other nodes during node joining. It includes the following fields: `scope_label` and `scope`.
- **CONTENT_REGISTRATION**: This method registers a particular content (*i.e.*, a FQDN or FQTN) to a scope. This registration allows nodes interested in a particular content to obtain a list of scopes updating (or consuming) that content. R-Nodes forward this message towards the Rvz-Node associated to the realm, which will generate R_CONTENT_REGISTRATION. It includes the following fields: `target_content`, `forwarding_scope_path_history`, `registered_scope`, `c_seq`, and `max_forwards`.
- **R_CONTENT_REGISTRATION**: Upon receiving an CONTENT_REGISTRATION request, the Rvz-Node responsible for the `target_content` will send an R_CONTENT_REGISTRATION response. It includes the following fields: `forwarding_scope_path`, `target_scope`, `source_scope`, `status`, `c_seq`, and `max_forwards`.

4.8.2 Resource location

In this section we describe the DCSIP methods for finding other nodes prior to exchange data-content with those nodes.

- **OBTAIN_SCOPE**: This method requests for a list of scopes bound to a particular FQDN or FQTN. This list enables I-Nodes for performing rendezvous with T-Nodes associated to the requested FQDN or FQTN. R-Nodes forward this message towards the Rvz-Node associated to the realm, which will generate R_OBTAIN_SCOPE. It includes the following fields: **target_content**, **forwarding_scope_path_history**, **source_scope**, **c_seq**, and **max_forwards**.
- **OBTAIN_SCOPE_SUBSCRIPTION**: This method creates a subscription to the rendezvous information of a particular FQDN or FQTN. An I-Node uses this procedure for keeping its information updated without flooding the network with OBTAIN_SCOPE messages. It includes the following fields: **target_content**, **forwarding_scope_path_history**, **source_scope**, **expiration**, **c_seq**, and **max_forwards**.
- **R_OBTAIN_SCOPE**: Upon receiving an OBTAIN_SCOPE request, the Rvz-Node responsible for the target_content will send an R_OBTAIN_SCOPE response. A Rvz-Node also sends this message to nodes subscribed to OBTAIN_SCOPE updates. It includes the following fields: **forwarding_scope_path**, **target_scope**, **source_scope**, **r_scope_list**, **status**, **c_seq**, and **max_forwards**.
- **OBTAIN_SCOPE_PATH**: This method requests for a scope-path (*i.e.*, a set of R-Nodes identified by a scope-label) which allows for reaching a particular target scope. It includes the following fields: **target_scope**, **forwarding_scope_path_history**, **source_scope**, **request_id**, **n_hops**, **c_seq**, and **max_forwards**.
- **R_OBTAIN_SCOPE_PATH**: Upon receiving an OBTAIN_SCOPE_PATH request, the T-Node responsible for the target_scope will send an R_OBTAIN_SCOPE_PATH response. It includes the following fields: **forwarding_scope_path**, **target_scope**, **source_scope**, **r_scope_path**, **status**, **c_seq**, and **max_forwards**.

4.8.3 Session establishment and maintenance

In this section we describe the DCSIP methods for creating and maintaining a DCSIP session between two nodes.

- **CREATE_SESSION (INVITE)**: This method requests for establishing a DCPS session between two scopes. It includes the following fields: **forwarding_scope_path**, **target_content**, **target_scope**, **source_scope**, **session_scope_path**,

session_heartbeat_period, **pdp_data**, **edp_data**, **session_id**, **c_seq**, and **max_forwards**.

- **ACCEPT_SESSION**: This method confirms the establishment of a DCPS session between two scopes. This method is usually sent by T-Nodes, but it can be also sent by a I-Node after receiving a **MODIFY_SESSION** message. It includes the following fields: **forwarding_scope_path**, **target_content**, **target_scope**, **source_scope**, **session_scope_path**, **session_heartbeat_period**, **pdp_data**, **edp_data**, **session_id**, **c_seq**, and **max_forwards**.
- **MODIFY_SESSION**: This method requests to amend a particular DCPS session. Currently, the only two parameters that are modifiable are the **session_scope_path** and the **session_heartbeat_period**. It includes the following fields: **forwarding_scope_path**, **target_content**, **target_scope**, **source_scope**, **session_scope_path**, **session_heartbeat_period**, **pdp_data**, **edp_data**, **session_id**, **c_seq**, and **max_forwards**.
- **CANCEL_SESSION**: This method cancels an existing session or rejects the establishment of a session between two scopes. It includes the following fields: **forwarding_scope_path**, **target_content**, **target_scope**, **source_scope**, **session_cancel_description**, **session_id**, **c_seq**, and **max_forwards**.
- **HEARTBEAT**: DCSIP nodes use this method for keeping a session alive. Additionally, it piggybacks information about end-to-end QoS violations and sample acknowledgments. It includes the following fields: **forwarding_scope_path**, **target_content**, **target_scope**, **source_scope**, **qos_status**, **ack**, **nack**, **session_id**, **c_seq**, and **max_forwards**.

4.9 DCSIP operation

In this section we describe the general operation of DCSIP through an example based on the health service use case of Section 4.2.

4.9.1 Joining and registration

The first step a node must perform is joining the system. R-Nodes joining consists of three steps:

1. Announcing their scope-label upstream.
2. Receiving their scope from a R-Node located upstream.

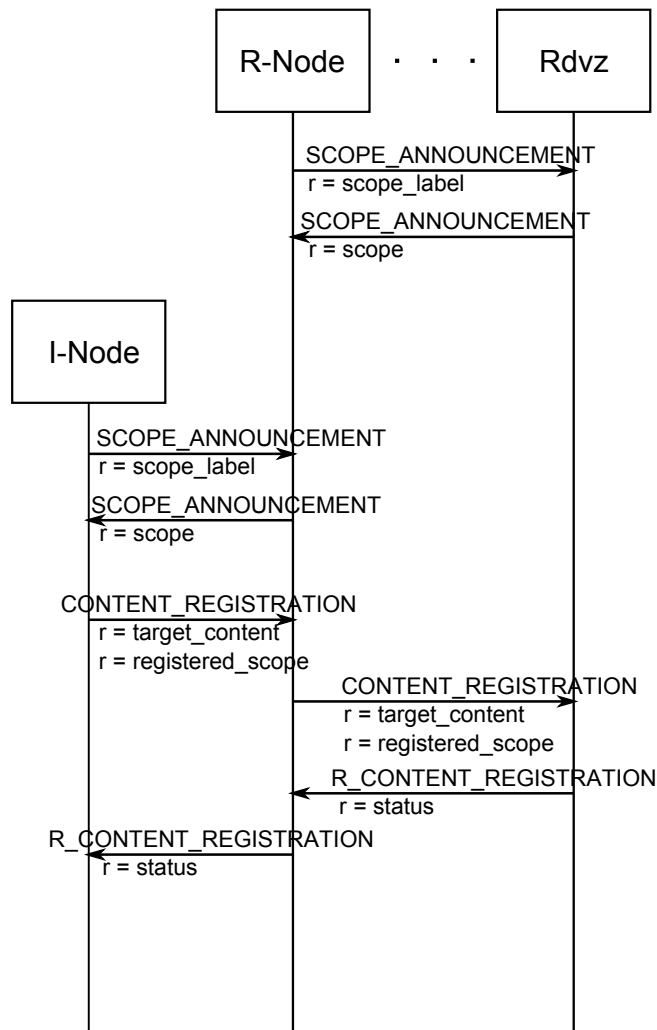


Figure 4.5: Node joining and content registering.

3. Announcing their scope downstream.

Endpoint nodes (I-Nodes and T-Nodes) does not own any scope-label, and therefore they just wait for receiving a SCOPE_ANNOUNCEMENT message from their upstream R-Node in order to know their scope.

Once a node knows its scope, it can register the content it wants to share by sending a CONTENT_REGISTRATION message to the Rdvz-Node. In our example, an I-Node associated to the medical records database of the *hvn* hospital registers the following information:

```
TARGET-CONT: dcps://sas.ddscloud.com/sas/medrecords/.par/
REG-SCOPE:  dcps://sas.ddscloud.com/hvn/meddb/
```

DESCRIPTION: "Hosp. Virgen de las Nieves - Records"

If the Rdz-Node accepts the registration, it will send back a R_CONTENT_REGISTRATION notifying the I-Node that the request was successfully processed. The process (sequence diagram) of joining the system and registering a content is depicted in Figure 4.5. In this figure, the relevant information that each node sends is referred to as $r=$.

4.9.2 Resource location and session establishment

Figure 4.6 depicts the sequence diagram for session creation in DCSIP. In this figure, the relevant information that each node sends is referred to as $r=$, whereas the information used for message routing is referred to as $d:$. The creation starts with an I-Node sending an OBTAIN_SCOPE message, which requests the list of scopes associated to a particular content (*i.e.*, a particular DCPS-Entity-Group). In particular, in our example the I-Node requests the following `target_content`:

TARGET-CONT: `dcps://sas.ddscloud.com/sas/medrecords/.par/`

Upon receiving this message, each R-Node in the network forwards it towards the Rvz-Node. This node controls a database that contains all the `target_content-registered_scope` associations.

The rendezvous node then processes the request and generates an R_OBTAIN_SCOPE message. This message contains a list of scopes that are associated to the requested content, and it is forwarded towards the I-Node using the `fwd_scope_path` field. In our example, the content of that list is:

TARGET-CONT: `dcps://sas.ddscloud.com/sas/medrecords/.par/`

SCOPE_LIST:

1. SCOPE: `dcps://sas.ddscloud.com/hch/medrec/`
DESCRIPTION: "Hosp. Carlos Haya - Records"
2. SCOPE: `dcps://sas.ddscloud.com/hvn/meddb/`
DESCRIPTION: "Hosp. Virgen de las Nieves - Records"
3. SCOPE: `dcps://sas.ddscloud.com/hsc/diag/`
DESCRIPTION: "Hosp. San Cecilio - Records"
4. SCOPE: `dcps://sas.ddscloud.com/htc/record/`
DESCRIPTION: "Hosp. Torrecardenas - Records"

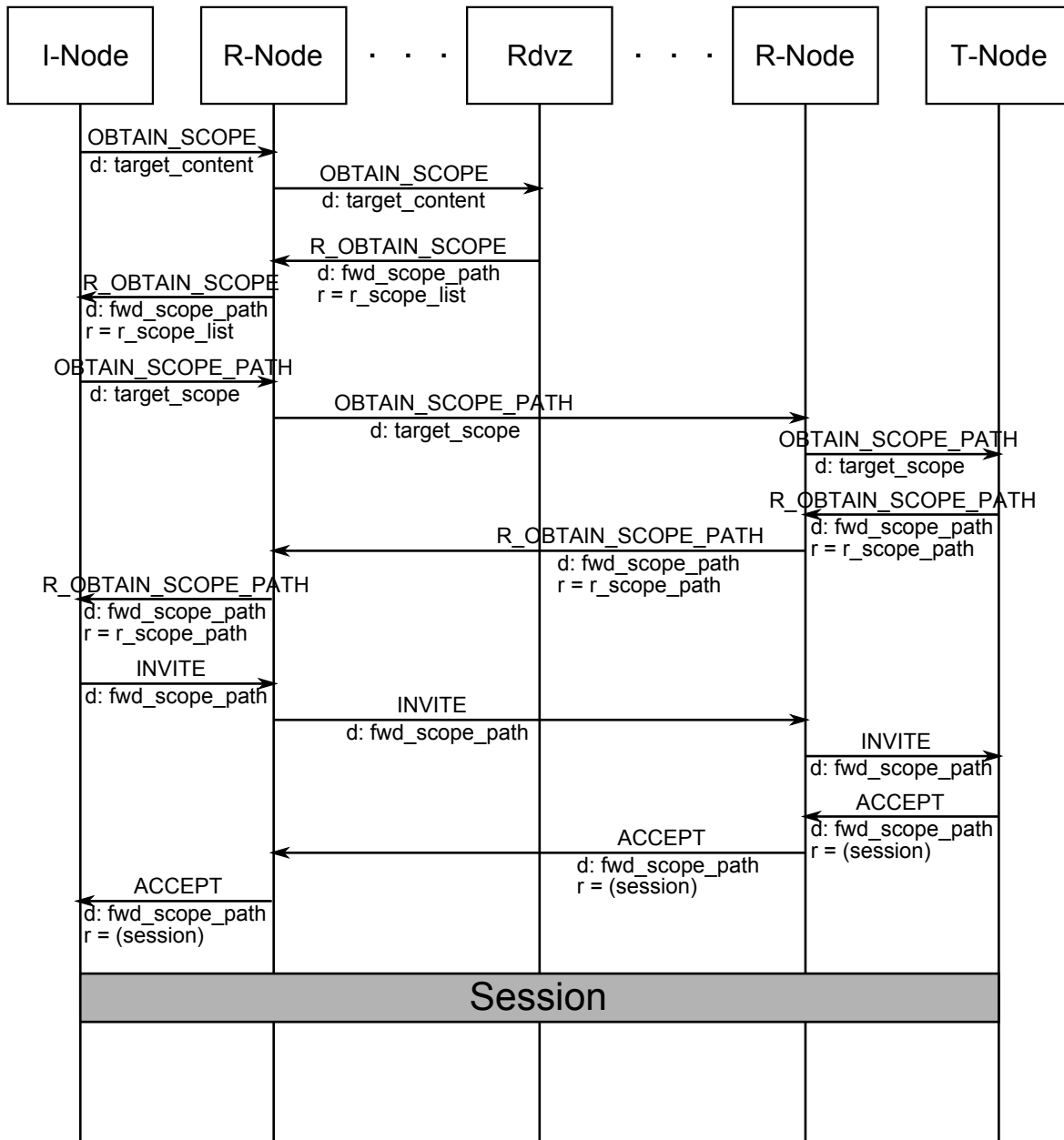


Figure 4.6: Session creation diagram.

Using the previous list, the I-Node can request for the scope-path associated to a particular scope. In our example, the application sends an OBTAIN_SCOPE_PATH message with the following scope:

TARGET_SCOPE: dcps://sas.ddscloud.com/hch/medrec/

R-Nodes forward the message towards the T-Node whose scope matches with the `target_scope` field. Upon reception, the T-Node sends an `R_OBTAIN_SCOPE_PATH` message back to the I-Node. This message includes an `r_scope_path` field. This field specifies a set of hops that a message sent from the I-Node can traverse to reach the T-Node. In our example, the content of the `r_scope_path` field is:

```
R_SCOPE_PATH:    dcps://meddb/hvn/hch/medrec/
```

After receiving the `R_OBTAIN_SCOPE_PATH` message, the I-Node knows a valid scope-path to the T-Node, and can start a session negotiation using an `INVITE` message addressed to that scope-path. This `INVITE` also includes a set of DCPS QoS parameters that both I-Node and T-Node must negotiate. If the T-Node accepts the session, it sends back an `ACCEPT` message. Upon receiving an `ACCEPT` message, each R-Nodes within the scope-path will create the proper route, which completes the session establishment.

A session will be active as long as the involved endpoint nodes receive `HEARTBEAT` messages or until one of them explicitly cancels the session. In addition, one of the communicating nodes can change session parameters by using a `MODIFY_SESSION` request.

4.10 SIP and DCSIP comparison

To complete the explanation of DCSIP, in this section we include Table 4.1. It compares one by one the differences between SIP and DCSIP. In particular, it contains general comparisons, together with message and field comparisons (and definitions).

Table 4.1: Comparison of SIP and DCSIP features.

SIP	DCSIP
General	
	Continuing next page

continued from previous page	
SIP	DCSIP
User Agent.	Node. Each node can contain one or more DDS participants.
User Agent Client.	I-Node.
User Agent Server.	T-Node.
Registrar.	Rdz-Node.
Proxy.	R-Node.
Content is addressed using UAs location .	Content is addressed using data-content name and DCPS-Entity-Group name.
Address-of-Record identifies a location .	FQDN , FQTN , and DCPS-Entity-Group identify data-content. For routing purposes, scope , scope_label , and scope_path identify a location.
SIP URI. It does not support DCPS entities.	A variation of CoAP URI that supports DCPS entities.
Typically, content consists of multimedia streams . The details of the session, such as the type of media, codec, or sampling rate, are not described using SIP. Rather, the body of a SIP message contains a description of the session, encoded in some other protocol format, such as sdp.	Content consists of DDS topics . The details of the session, such as the topic name, topic type, or QoS policies, are described using PDP and EDP.
	Continuing next page

continued from previous page	
SIP	DCSIP
Typically, content sent over RTP.	Content sent over DDS-RTPS.
Message comparison	
There are no methods that allow nodes to automatically build their URI according to their location in the system.	DCSIP defines a method for automatically obtain the scope of a node. Since each node can automatically (<i>i.e.</i> , without human intervention) re-calculate its scope after a change in realm's topology, this eases the evolution of DCSIP-based systems. SCOPE_ANNOUNCEMENT.
REGISTER method for registering a SIP URI to a user.	CONTENT_REGISTRATION method for registering a scope to a content.
There are no methods for performing resource location. Resources (UAs) are located automatically during invitation. Consequently, calling to a specific device is not supported.	DCSIP defines methods for locating nodes that share a particular data-content. Methods for obtaining a list of scopes associated to a given content: OBTAIN_SCOPE. OBTAIN_SCOPE_SUBSCRIPTION. Method for obtaining the scope_path (<i>i.e.</i> , the route) to a given scope: OBTAIN_SCOPE_PATH.
	Continuing next page

continued from previous page	
SIP	DCSIP
<p>Method for creating, modifying, and maintaining a session:</p> <p>INVITE.</p> <p>Response for accepting a session:</p> <p>200 Ok.</p>	<p>Methods for creating, accepting, modifying, and maintaining a session:</p> <p>CREATE_SESSION.</p> <p>ACCEPT_SESSION.</p> <p>MODIFY_SESSION.</p> <p>HEARTBEAT.</p>
<p>Methods for session cancelation:</p> <p>BYE.</p> <p>CANCEL.</p>	<p>Method for session cancelation:</p> <p>CANCEL_SESSION.</p>
<p>There are six classes of responses according to the obtained result:</p> <p>1xx: Provisional.</p> <p>2xx: Success.</p> <p>3xx: Redirection.</p> <p>4xx: Client Error.</p> <p>5xx: Server Error.</p> <p>6xx: Global Failure.</p>	<p>There are three classes of responses according to the original request. The result is indicated in the status field:</p> <p>R_CONTENT_REGISTRATION.</p> <p>R_OBTAIN_SCOPE.</p> <p>R_OBTAIN_SCOPE_PATH.</p>
	Continuing next page

continued from previous page	
SIP	DCSIP
Field comparison	
Request-URI: This header identifies the domain of the location service for which the registration is meant (e.g., "sip:chicago.com"). The "userinfo" and "@" components of the SIP URI must not be present.	Not used.
Not used.	Fields specific to SCOPE_ANNOUNCEMENT : scope_label: The scope-label associated to the node. scope: The scope associated to the node.
To: This field contains the address-of-record whose registration is to be created, queried, or modified. The To header field and the Request-URI field typically differ, as the former contains a user name. This address-of-record must be a SIP URI or SIPS URI.	target_content: A FQDN, FQTN, or DCPS-Entity-Group URI that identifies the data-segment to share. target_scope: The scope associated to the T-Node. R-Nodes use this field for performing access control checks. T-Nodes use this field for checking if the message is addressed to them.
From: This field contains the address-of-record of the person responsible for the registration. The value is the same as the To header field unless the request is a third-party registration.	source_scope: The scope associated to the I-Node. R-Nodes and T-Nodes use this field for performing access control checks.
	Continuing next page

continued from previous page	
SIP	DCSIP
<p>Call-ID: All registrations from a UAC should use the same Call-ID header field value for registrations sent to a particular registrar.</p>	<p>session_id: This is a unique hash that identifies a session negotiation request across the realm.</p>
<p>CSeq: The CSeq value guarantees proper ordering of requests. A UA must increment the CSeq value by one for each request with the same Call-ID.</p>	<p>c.seq: This field identifies and orders transactions. A I-Node must increment c.-seq value by one for each request with the same session_id.</p>
<p>Record-Route (<i>optional</i>): The Record-Route header field is inserted by proxies in a request to force future requests in the dialog to be routed through the proxy.</p>	<p>forwarding_scope_path_history: This field contains a scope-path describing the R-Nodes that the message has already traversed. This is used as a reverse route for delivering the message's response. In order to achieve this behavior, R-Nodes add their scope-label to this field before forwarding the message.</p>
<p>Via: The Via header field indicates the transport used for the transaction and identifies the location where the response is to be sent.</p>	<p>r_scope_path: This field contains a scope-path describing the sequence of R-Nodes that a message from I-Node has to traverse in order to reach the target_scope. Initially, this field is set with a reversed version of the forwarding_scope_path_history field. This field can not be modified by intermediate R-Nodes.</p> <p>forwarding_scope_path: This field contains a scope-path describing the set of R-Nodes the message has to traverse in order to reach a particular I-Node. This field is used as the route for delivering the message. In order to determine the next hop, R-Nodes remove their scope-label from the scope-path before forwarding the message.</p> <p>session_scope_path: This field contains the sequence of R-Nodes that I-Node is using for reaching the T-Node. If the T-Node confirms the session, the session will be established using this scope-path.</p>
	Continuing next page

continued from previous page	
SIP	DCSIP
<p>Max-Forwards: This field limits the number of hops a request can transit on the way to its destination. It consists of an integer that is decremented by one at each hop. If the Max-Forwards value reaches 0 before the request reaches its destination, it will be rejected.</p>	<p>max_forwards: This field contains the maximum number of hops the message can traverse before being discarded. It consists of an integer that is decremented by one at each hop. If the max_forwards value reaches 0 before the request reaches its destination, it will be rejected.</p>
<p>Expires (<i>optional</i>): This field specifies the relative time after which the message (or content) expires.</p>	<p>session_heartbeat_period: This field specifies the relative time after which the session expires. The value of the parameter is an integer indicating a number of seconds.</p>
<p>Body (<i>optional</i>): This field contains the parameters of the session. Typical parameters are used ports and supported codecs.</p>	<p>pdp_data: The participant discovery information. This field contains data associated to SPDP or another alternative discovery protocol.</p> <p>edp_data: The endpoint discovery information. Among other information, it includes the <code>DataRepresentationQosPolicy</code>, which describes the used data-types. It also includes QoS policies that are enforced in an end-to-end basis. This field contains data associated to SEDP or another alternative discovery protocol.</p>
	Continuing next page

continued from previous page	
SIP	DCSIP
<p>Contact: This field includes zero or more values containing address bindings. Contact contains a SIP or SIPS URI that represents a direct route to contact Alice, usually composed of a username at a FQDN.</p>	<p>Field specific to CONTENT_REGISTRATION:</p> <p>registered_scope: This field includes one address binding. It contains a URI that represents a direct route to contact the registered node (<i>i.e.</i>, the node's scope). It can optionally contain a scope description (a string describing the scope). It can also optionally contain a list of DCPS-Entity-Names.</p>
Not used.	<p>Field specific to responses:</p> <p>status: This field indicates if the operation was successfully completed or not. In case of an error, it indicates the causes.</p>
Not used.	<p>Field specific to OBTAIN_SCOPE_SUBSCRIPTION:</p> <p>expiration: This field specifies the duration of the subscription. The value of the parameter is an integer indicating seconds.</p>
Not used.	<p>Fields specific to OBTAIN_SCOPE_PATH:</p> <p>request_id: This is a unique hash that identifies an OBTAIN_SCOPE_PATH request across the realm. This enables to avoid overflowing the system with OBTAIN_ROUTE messages.</p> <p>n_hops: The number of R-Nodes an OBTAIN_SCOPE_PATH message has traversed. When a new OBTAIN_SCOPE_PATH message arrives to an R-Node, the values of its n_hops and request_id fields are stored. When a subsequent OBTAIN_SCOPE_PATH message arrives to the same node, the R-Node compares the new n_hops field with the stored value. If the new value is equal or greater than the stored value, the message is discarded. In this way, messages associated to non optimal routes (<i>i.e.</i>, the ones that require more hops) do not overflow the system.</p>
	Continuing next page

continued from previous page	
SIP	DCSIP
Not used.	Field specific to R.OBTAIN.SCOPE : r_scope_list : The list of scopes containing entities interested in the target FQDN/FQTN. This field can not be modified by intermediate R-Nodes.
Not used.	Field specific to CANCEL_SESSION : session_cancel_description : This field describes cancelation's causes.
Not used.	Fields specific to HEARTBEAT : qos_status : This field contains information about QoS violations. ack : This field specifies the next expected sample sequence number. nack : This field contains a list of samples not received.

From this table we can conclude that DCSIP is more suited than SIP for the unique requirements of DCPS scenarios.

First, DCSIP addresses content using a name that is not bound to any specific location, whereas SIP uses the UA location. In addition, DCSIP supports DCPS entities and have mechanisms for univocally identifying DCPS content, which are not supported by SIP.

DCSIP messages are routed using `scope`, `scope_label`, and `scope_path`, which are concepts well aligned with the NDN model (see Section 6.5), whereas SIP messages are routed by using an IP address or name associated to a particular location. In this regard, DCSIP provides mechanisms for automatically creating an hierarchical architecture for routing data. This architecture is able to adapt to the changes of the system topology.

Finally, DCSIP have mechanisms for discovering all the publications and subscriptions associated to a particular topic or data-space, which is not supported by SIP.

4.11 DDS and DCSIP features comparison

Finally, in this section we compare the differences between current DDS standard family and the proposed DCSIP. More precisely, Table 4.2 includes discovery and signaling differences and similarities.

Table 4.2: Comparison of current DDS and DCSIP features.

DDS	DCSIP
Discovery	
DDS-RTPS defines a mandatory-to-implement discovery protocol, the SDP. DDS-RTPS allows implementations to use alternative discovery protocols.	Regarding discovery, DCSIP is compatible with DDS-RTPS, since it supports exchanging SPDP and SEDP information as a parameter of the session.
DDS defines the domainId for identifying DDS domains. SDP uses the domainId (an integer) to calculate the ports that DDS uses. domainId is not valid for universally identifying DDS domains.	DCSIP defines FQDN for identifying DDS domains. FQDN is a URI that consists of a DNS namespace associated to a particular realm and a data-space name. FQDN universally identifies DDS domains.
DDS uses topic name and type name for identifying DDS topics. DDS applications that use SDP exchange this information through SEDP. topic name and type name are not valid for universally identifying DDS domains.	DCSIP defines FQTN for identifying DDS topics. FQTN is a URI that consists of a FQDN (associated to topic's data-space) and a topic name. FQTN universally identifies DDS topics.
	Continuing next page

continued from previous page	
DDS	DCSIP
<p>DDS-RTPS defines the GUID, a sequence of 16 bytes, for univocally identifying DDS entities within a data-space.</p> <p>DDS-RTPS GUID is not human readable, and it can not include descriptive information for an entity. Consequently, GUID is not suited to create lists of sensors and actuators that are readable and meaningful for human administrators.</p>	<p>DCSIP defines DCPS-Entity-Name for identifying DDS entities. DCPS-Entity-Name is a URI that univocally identifies a participant, publisher, or subscriber within a scope.</p> <p>DCPS-Entity-Name can include a meaningful name that helps human administrators to identify the associated entity (<i>e.g.</i>, <code>./temperature-sensor01</code>).</p>
<p>Since existing identifiers for domains, topics, and entities are not universal, it is not possible to discover all the DDS entities associated to a particular topic or domain in a WAN environment.</p>	<p>DCSIP allows DDS applications to obtain a list with all the participants, publishers, or subscribers associated to a particular data-space or topic. This list is stored in a Rdz-Node.</p>
<p>DDS-RTPS does not define a specific addressing scheme for describing entities location.</p>	<p>In addition to FQDN, FQTN, and DCPS-Entity-Name, which identify content, DCSIP defines scope and scope_label for univocally identifying the location of a DDS entity in the realm.</p>
Signaling	
<p>DDS sessions are not standardized.</p>	<p>DCSIP is a DDS session control protocol.</p> <p>DDS session defines a common state where applications share data-content. This common state is characterized by a set of QoS.</p> <p>Applications use DCSIP to negotiate the characteristics of DDS sessions (topics, QoS policies).</p>
	<p>Continuing next page</p>

continued from previous page	
DDS	DCSIP
<p>DDS-IS is not standardized, therefore DDS does not define mechanisms for negotiating the creation of DDS-IS routes.</p>	<p>DCSIP allows DDS applications to negotiate the creation of DDS-IS routes.</p> <p>Using DCSIP, applications can select the optimal route for sharing data-content.</p> <p>Since DDS-IS nodes do not propagate DDS discovery and data-content information until two endpoints have declared their intention to share data-content, DCSIP helps to increase the scalability of DDS-IS based systems.</p>

4.12 Conclusions

In this chapter we have proposed the the rationale and fundamentals of a protocol for session signaling in DCPS environments. This protocol allows applications to perform DCPS entity and data-content discovery in medium-scale environments. It also allows the creation of sessions between nodes, which eases the creation of DCPS routes and enables the enforcement of QoS at end-to-end level.

The protocol has not yet been implemented nor integrated with the DDS-IS at the time of writing. However, one of the main implementors of DDS, RTI [89], has already shown its interest in DCSIP implementation, which envisages a very promising future for the proposal.

In spite of its benefits, the proposed solution is not suitable for very large DCPS environments, as the presence of a unique rendezvous server may not scale for very high-scale scenarios (deployments with thousands of nodes). Moreover, DCSIP signaling is conceived for working in a DDS/DDS-IS-based scenario, and it does not contain information for its routing in non-DDS scenarios. For overcome these issues, in the next chapter we propose a solution based on P2P-SIP/RELOAD for resource discovery and connection establishment between DCPS nodes.

DDS usage for RELOAD

IN the previous chapter we defined an information model for deploying DCPS system in medium-scale scenarios. We also proposed a protocol for resource discovery and session establishment in medium-scale scenarios.

In this chapter we focus in solving a still open issue in DCPS environments: their scalability in large-scale deployments. In particular, and taking advantage of the resource naming format proposed in the previous chapter, we define an extension to the RELOAD standard [44] for providing resource discovery and NAT traversal in DCPS environments.

To motivate the need of solving the scalability issues of DCPS for large-scale deployments, we study a set of use cases that are based on the concept of Internet of Things (IoT).

5.1 Motivation

The IoT [5] has been gaining momentum in the last years. The reduction of technology costs and the great market penetration of handsets have allowed people to stay permanently connected to the Internet. In this context –where people already make an extensive use of the Internet–, the next step is to connect every device that generates relevant information to the Cloud [10], such that any object becomes smart and accessible.

In this new approach, users and applications will be able to interact among

themselves and their intelligent environment. For instance, users will receive alerts about expired food from their fridge, or will check whether their garage door is closed; it will even be possible for coffee machines to automatically prepare the coffee in the morning when the alarm goes off.

The core of IoT is the data-content shared by non-traditional-Internet devices, such as appliances, doors, alarm-clocks, cars, etc. These devices can generate a lot of data-content –*e.g.*, temperature updates, device status reports, car metrics, and many others– Hence, IoT deployments demand efficient and robust communications with special scalability support.

IoT scenarios are especially suited for taking advantage of DCPS approaches. As we studied in Section 2.2.1, DCPS is a programming model that is not focused on data location, but instead on the data-content itself. In DCPS deployments, data-content flows between data sources (*i.e.*, publishers) and data sinks (*i.e.*, subscribers).

Publishers and subscribers access to data-content in a DCPS basis. That is, instead of requesting for a resource location, they create a publication (or a subscription) to a segment of a data-space (*i.e.*, a topic). Topics are decoupled spatially, and therefore subscribers can access to topic's content without a prior knowledge of publishers' location.

Another advantage of DCPS systems is that they are not data-agnostic, on the contrary DCPS allows for performing advanced operations over data-content, such as content-based filtering, data transformation, and data composition.

However, prior to sharing data, publishers and subscribers must discover each other (*i.e.*, a rendezvous function is needed). This rendezvous function has to support multiple publishers and subscribers sharing a particular topic from several different locations.

A simple solution for small LAN deployments is to use multicast for discovering publishers and subscribers. However, this solution does not scale well and is not suitable for environments with NATs and firewalls.

Another possible solution is to set up a server for keeping all the rendezvous information, as the one proposed in Chapter 4. The use of a properly configured server avoids the NAT traversal problem for discovery, but does not scale in large-scale deployments, does not solve NAT for information exchange, and introduces a single point of failure.

A third solution is to distribute the discovery information among the members of a P2P overlay. P2P deployments are highly scalable because they distribute computing and traffic load among all the members of the overlay. In addition, some P2P solutions support advanced functionalities like NAT traversal. One of these

solutions is RELOAD [44], an IETF specification for building and maintaining P2P systems on the Internet. RELOAD provides an abstract overlay service that can be extended to support specific application usages. For more information about RELOAD, please refer to Section 2.4.

In this chapter we propose a scalable RELOAD-based solution to the discovery problem in DCPS environments. Our proposal demonstrates the extensibility of RELOAD by proposing a RELOAD usage that adapts it for performing rendezvous in DCPS-based deployments. We define the Resource Naming Format, the data types for storing the rendezvous information, and the interactions among the entities performing the rendezvous.

Once two particular nodes complete the rendezvous, they can establish a direct connection by means of existing RELOAD mechanisms. Using this connection, nodes can exchange IoT data-content using a protocol for constrained nodes, such as CoAP [97]; or a DCPS middleware, such as the OMG DDS [67].

We have implemented a proof of concept prototype and conducted a set of experiments over a simulated network of 10000 nodes. Our results show that the average and 95 percentile latency for storing and retrieving the discovery information of a DCPS follow a logarithmic growth as a function of network size.

The remainder of the chapter is organized as follows. In Section 5.2 we present a set of motivating use cases. In Section 5.3 we identify the limitations of RELOAD we need to overcome. In sections 5.4 and 5.5 we study the system design and operation. In sections 5.6 and 5.7 we address the experimental setup and the conducted experiments, and we analyze the obtained results. Finally, in Section 5.8 we summarize the main conclusions of the chapter.

5.2 Motivating use case examples

In this section we provide a set of use cases we considered when designing the RELOAD usage for DCPS-based IoT systems. Each use case requires performing rendezvous with a different set of DCPS entities.

5.2.1 User data-space

In this use case, a user is able to obtain information of his "user data-space", which is the set of data related to him.

A typical user data-space contains information about user's vitals, car position, gas remaining at his car, status of his fridge, and status of user's house doors. All

this information is relevant to a particular user, and should be accessible only by him.

This use case will allow a user to discover and subscribe to any piece of information within his data-space. Consequently, it is necessary to discover both publishers and subscribers associated to certain data-space, without restrictions on the types of exchanged data-content.

5.2.2 N permanent stock-shares publishers, M variable stock-shares subscribers

In this scenario, a constant number of stock-shares publishers publish information of share prizes to a data-space. A variable number of subscribers (users who want to obtain the latest share values) subscribe to this information.

This scenario requires subscribers to be able to discover stock-shares publishers associated to a certain market. Once discovered, the subscriber is able to subscribe to the publishers of interest.

5.2.3 N variable temperature sensors, 1 permanent temperature monitor

In this use case, a set of temperature sensors publish temperature values to a data-space. These values must reach a temperature monitor that is subscribed to the temperature updates. A temperature monitor is a node that gathers and stores temperature values and may generate alerts. These alerts could be used for triggering automatic responses or they can just generate a notification to a human operator. A temperature monitor node is always available, but new temperature-publishers may be added later.

This scenario requires the temperature monitor to be able to subscribe to temperature sensors, even if they are not yet active. In this sense, once a new temperature sensor joins the system, it will discover the temperature monitor that is subscribed to its temperature updates.

5.2.4 N variable planes (entering and leaving the airspace), M variable air traffic controllers

In this use case, an Air Traffic Controller (ATC) receives the information of planes flying in his Flight Information Region (FIR). As planes enter and exit the FIR, their

associate publishers (in this example, with information about the sensors in the plane) join and leave respectively the FIR data-space. Additionally, an authorized ATC could join latter to the data-space, so the system should deliver the existing planes information to this new ATC.

This use case requires both publishers (planes) and subscribers (ATCs) to be able to discover already existing data-space participants. As an additional constraint, the information about existing publishers (planes) should be visible only to the authorized ATCs. Similarly, information about subscribers should be only visible to allowed publishers.

5.3 New required features

In Section 2.4.5, we studied the existing RELOAD usages. In all of those usages, the system was user- or service-centric. That is, it stores the location of a particular user agent or of a peer providing a service.

However, DCPS systems –and specifically the use cases described in the previous section–, require the following functionality not currently supported by RELOAD:

- **Discovery of multiple publishers** updating a particular topic. This feature will allow subscribers to perform rendezvous with compatible publishers.
- **Discovery of multiple subscribers** interested in a particular topic. This feature will allow publishers to locate subscribers interested in their data updates.
- **Discovery of multiple data-participants** associated to certain data-space. This feature will allow applications for locating other applications associated to a particular data-space (*i.e.*, to a collection of topics).

5.4 System design

In the this section we study the characteristics of the proposed architecture and the adopted design decisions.

5.4.1 A DCPS approach

The proposed design has been developed around the DCPS model already studied in Section 2.2.1.

Since our proposal is data-centric, the relation between nodes and overlay is based on the data that they publish or subscribe. Consequently, nodes should be discoverable only if they have some piece of data to share. However, all the nodes sharing information do not necessarily had to be discoverable (see use cases in Sections 5.2.2 and 5.2.3).

Note that with "discoverable" we mean that a given node is available for rendezvous at the application level. Therefore, this has no impact on maintenance or routing operations of the underlying layers.

5.4.2 A RELOAD based architecture

We have selected RELOAD for performing the rendezvous among DCPS entities because it provides high scalability, robustness, and NAT traversal. High scalability is a requirement, as our system must be suitable for large-scale deployments. NAT traversal is also necessary, as it allows administrators to deploy our architecture in WAN environments that include NATs and firewalls. Robustness is also a main concern, as we assume that nodes in the overlay can fail, and a single point of failure is not admissible.

Figure 5.1 depicts the proposed layer architecture, which is divided in three layers: RELOAD, Discovery-and-Data, and DCPS application. This architecture adopts a layered design based on the following motivations. First, it increases system modularity, by allowing the specific implementation of each layer to be changed without impacting the rest of the layers. Second, it allows existing applications to remain independent of new proposed entities while benefiting from the system. And third, it simplifies the overall understanding of the system, because it classifies entities of different functionality in different layers.

The bottom layer is named RELOAD, and contains RELOAD nodes (peers and clients). In order to identify these nodes, each node has a unique Node-ID, obtained during the process of joining the overlay (see Section 5.5.1). Node-ID allows RELOAD to locate and deliver messages to any member of the overlay infrastructure.

On top of RELOAD we define the Discovery-and-Data layer. This layer enables publishers, subscribers, and data-participants at the DCPS Application layer for performing rendezvous and for exchanging data-content. Discovery-and-Data layer is populated by D-Nodes and S-Nodes, both defined in Section 5.4.3. As we will see in Section 5.4.4, these nodes form groups according to the DCPS Application layer entities that they host.

Finally, the DCPS Application layer contains the DCPS entities (publishers, sub-

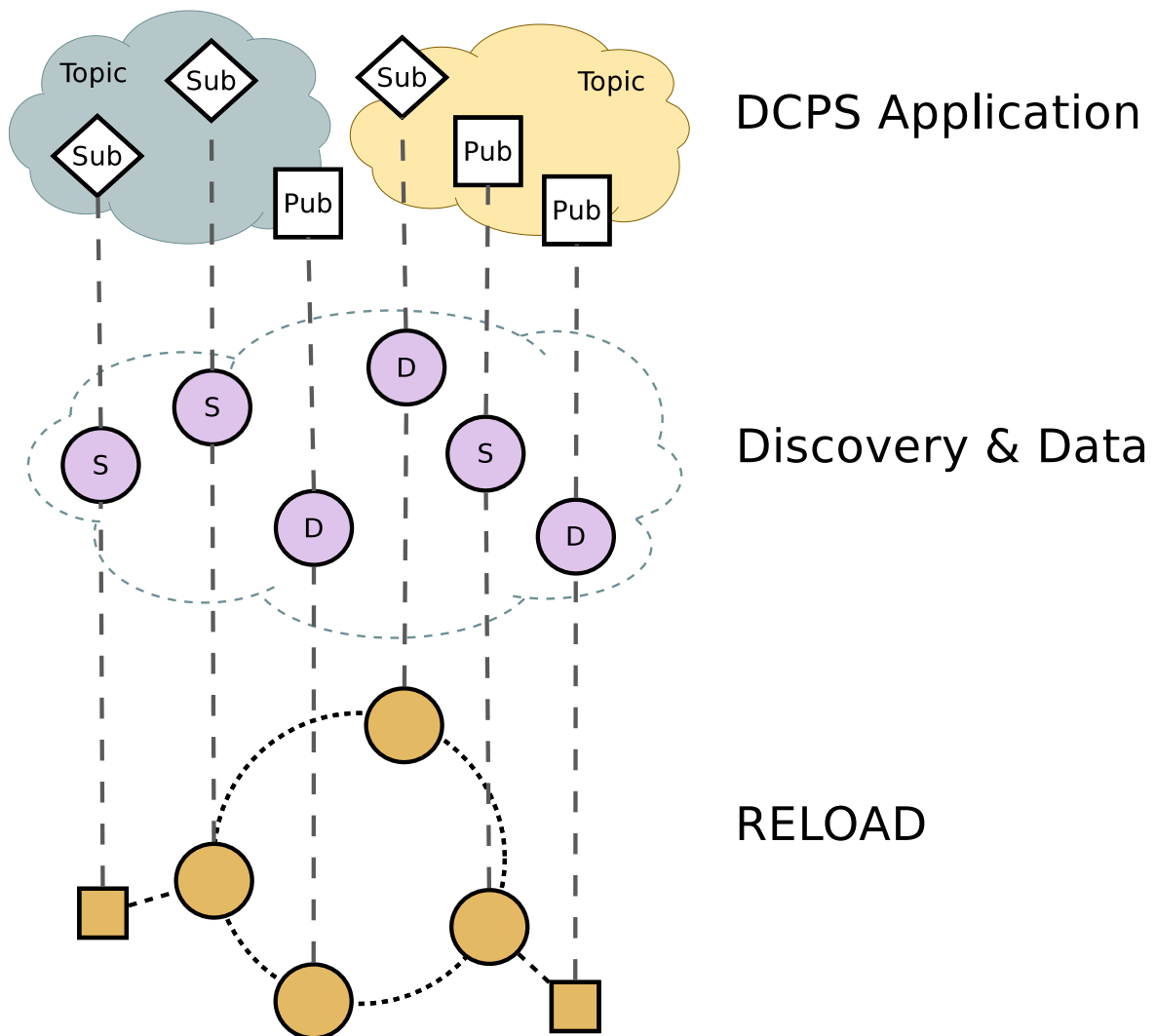


Figure 5.1: Architecture layers.

scribers, and data-participants) that exchange data-content using a DCPS approach.

As part of our design, we propose a new RELOAD usage that is adapted to the unique requirements of DCPS systems. This new DCPS usage for RELOAD consists of three different aspects:

1. The creation of new Kinds that respond to DCPS system requirements, proposed in Section 5.4.5.
2. The specification of a format for addressing data-content, described in Section

5.4.6.

3. The description of the usage operation, detailed in Section 5.5.

5.4.3 Discovery-and-Data node roles

Our proposal defines two different node roles for the Discovery-and-Data-layer. These roles cover two different relations between nodes and the overlay.

- **Discoverable Node (D-NODE)**: A member of the overlay that is visible to other nodes. It can act as subscriber, publisher, or data-participant. Since its rendezvous information is available to the nodes of certain DCPS Entity Group (see Section 5.4.4), those nodes can dialogue with it.

- **Spectator Node (S-NODE)**: A member of the overlay that does not share its presence and it is not discoverable. Consequently, S-NODEs can access D-NODEs discovery information, but their own information is not visible to other nodes. S-NODEs may eventually become a D-NODE (and *vice versa*).

These roles are independent of RELOAD node types: clients and peers can both act as D-NODEs or as S-NODEs. These roles are also independent of the hosted DCPS entities: D-NODEs and S-NODEs can both host publishers, subscribers, and data-participants.

5.4.4 DCPS Entity Groups

The proposed RELOAD usage takes advantage of the concept of DCPS-Entity-Group, defined in Section 4.5.4. In the context of this usage for RELOAD, DCPS-Entity-Group is an identifier shared among a group of Discovery-and-Data-layer nodes that host a particular group of DCPS Application layer entities.

We use DCPS-Entity-Group to determine if a given node is authorized to complete a store over a particular Resource-ID, as we study in next sections.

5.4.5 RELOAD Kinds for DCPS

S-NODEs can turn into D-NODEs (and *vice versa*). Once a S-NODE turns into a D-NODE, it becomes discoverable by a particular set of nodes.

Specifically, for dealing with the use cases in Section 5.2, we identify three different discovery types:

- **Topic publisher discovery:** This is the procedure of discovering all the nodes associated to certain topic as publishers (*i.e.*, those nodes that publish data of certain type).

- **Topic subscriber discovery:** This is the procedure of discovering all the nodes interested in certain topic (*i.e.*, those nodes subscribed to data of certain type).

- **Data-space discovery:** This is the procedure of discovering nodes running applications that interact with a certain data-space. The discovered nodes can contain multiple publishers and/or multiple subscribers of different data types.

In our system, the conversion between S-NODE and D-NODE is performed through a DataRegistration. We use DataRegistration for enabling discovery and for starting a publication or a subscription. A DataRegistration is a RELOAD store request of a specific RELOAD Kind (see Section 2.4).

In the following sections we propose three new RELOAD Kinds that constitute a new RELOAD Usage for DCPS systems. These new Kinds cover the three different discovery types we have described above.

5.4.5.1 TopicPublisher and TopicSubscriber Registrations

Topic publisher discovery and topic subscriber discovery are respectively the procedures of locating publishers and subscribers of a particular topic. In our system, these discovery types are accomplished by means of TopicPublisher and TopicSubscriber Registrations.

TopicPublisher Registration is a new RELOAD Kind for storing lists of publishers. Specifically, each list contains all the nodes declared as publishers of a particular topic. Any node that is able to access to a particular TopicPublisher Registration will also be able to perform rendezvous with the publishing nodes contained in that Registration. In this sense, a node that creates a TopicPublisher Registration entry is implicitly allowing a group of subscribers for subscribing to the publishers associated to that node.

TopicSubscriber Registration is a new RELOAD Kind for storing lists of subscribers. A node that registers into a TopicSubscriber Registration is implicitly creating a subscription to a topic, and therefore is allowing topic publishers for delivering topic updates to the subscribers associated to that node.

One of the benefits of this approach is that nodes can create an entry to a particular TopicSubscriber Registration even if there is not any active publisher for the associated topic. In this way, as soon as a publisher joins the overlay, it can perform rendezvous (and start sending topic updates) using the TopicSubscriber Registration information.

We have adopted RELOAD dictionary [44] (see Section 2.4.4) as the data model for storing TopicPublisher and TopicSubscriber Registrations. The dictionary model allows for univocally identifying each list entry with a key. In our design, the dictionary key for an entry is the Node-ID of the storing node. This characteristic avoids the duplication of entries, as each node can have at most one entry stored for a particular TopicPublisher (or TopicSubscriber) Registration.

Regarding content addressing, each dictionary record is univocally identified by a Resource-ID. This Resource-ID identifies the topic and type (publisher or subscriber) of the DCPS entities stored in the dictionary. We will explain how the Resource-ID is calculated in Section 5.4.6.

As we have explained above, each node can store at most one entry per dictionary record. However, there are cases where a given node contains multiple DCPS entities of the same type and data-segment (*e.g.* a node that contains multiple temperature publishers). In order to univocally identify the entities associated to a particular node, each dictionary entry contains a list of DCPS-Entity-Names (see Section 4.5.4), each one associated to an entity.

In order to preserve the integrity and authenticity of the information, the overlay processes a given store request only if the storing node's certificate has an adequate Node-ID (*i.e.*, the same as the one used as storing key), and if the certificate contains a DCPS-Entity-Group that matches with the Resource-ID of the store request.

If the application requires privacy, nodes can encrypt stored data by using a key associated to the DCPS-Entity-Group. In this way, only nodes belonging to the right DCPS-Entity-Group will be able to successfully read the content of a given dictionary record.

5.4.5.2 DataSpaceParticipant Registration

In addition to publisher and subscriber discovery, our system supports rendezvous among applications bound to a particular data-space. These applications can interact with multiple topics of a given data-space, using multiple publishers and/or subscribers.

DataSpaceParticipant Registration enables nodes for performing rendezvous with all the discoverable nodes that host applications associated to a particular data-space. In this way, applications do not have to create multiple TopicPublisher and TopicSubscriber Registrations, instead they access to a unique DataSpaceParticipant Registration.

DataSpaceParticipant is especially useful for deployments where one or more monitors access to data associated to several different topics (an example of this

scenario is the use case described in Section 5.2.1).

DataSpaceParticipant Registration is similar to TopicPublisher/TopicSubscriber Registrations. However, it includes a DCPS-Entity-Descriptor associated to each DCPS-Entity-Name to indicate if the entity is a publisher, a subscriber, or a data-participant.

DataSpaceParticipant uses the same mechanisms as the TopicPublisher and TopicSubscriber Registrations for providing integrity, authenticity, and privacy.

5.4.6 Resource naming format

In RELOAD, stored information is accessed by its Resource-ID. In this section we explain how our system calculates the Resource-ID for each DataRegistration type.

In our proposal, a Resource-IDs is generated from an URI that identifies certain data-space segment. The format for this URI is very similar to the one proposed in Section 4.6.

In particular, in our RELOAD usage the URI for a given FQDN is built as follows:

```
dcps:// < overlay > / < data-space-name > /
```

Where `dcps` stands for Data-Centric Publish-Subscribe, `overlay` is a DNS domain name identifying a P2P overlay, and `data-space-name` could be a hierarchy of data-space names (separated by forward slashes), for example:

```
dcps://ugr.es/labs/microwaves/
```

Since a topic is a segment of a data-space, the URI for a FQTN is built by attaching the topic-name to the FQDN of the topic's data-space:

```
dcps:// < overlay > / < data-space-name > / < topic-name > /
```

The URI associated to TopicPublisher, TopicSubscriber, and DataSpaceParticipant Registration is a DCPS-Entity-Group URI. In this regard, nodes build TopicPublisher, TopicSubscriber, and DataSpaceParticipant Registration by adding `/.pub/`, `/.sub/`, or `/.par/` suffixes to a FQDN or FQTN URI.

Our design also uses this URI format for the DCPS-Entity-Names (see sections 4.5.4 and 5.4.5.1). In order to save storage space and traffic load, our system stores DCPS-Entity-Names as relative URIs (using the dictionary record's URI as base).

5.5 System operation

In this section we describe the operation of our system through an example. In particular, we will use user data-space use case from Section 5.2.1. We have chosen this use case because it covers all the different DataRegistration types.

In this example we assume that RELOAD uses Chord [102] [103] as DHT algorithm. Chord uses SHA-1 [46] as the base hash function for assigning each node and resource a unique m -bit identifier (respectively called Node-ID and Resource-ID). These identifiers are used for creating a ring of 2^m nodes, which is called the Chord ring.

Each member of the ring maintains a routing table consisting of a finger table and a neighbor table, which allows nodes for locating resources on the overlay. The neighbor table contains a list of node's immediate successors and predecessors. The finger table stores information about $O(\log N)$ nodes distributed along the ring. Specifically, in Chord the i th entry in the finger table at node n contains the identity of the first node, s , that succeeds n by at least $2^{(i-1)}$ on the ring. In RELOAD, finger table entries are indexed in opposite order (*i.e.*, `finger[0]` is the node 180 degrees around the ring from the original node).

As we have no made modifications to the underlying Chord DHT, in this section we will describe the operation from a RELOAD-level point of view. For more information about how RELOAD functions are mapped to Chord DHT, please refer to RELOAD specification [44].

5.5.1 Joining the overlay

Prior to joining an overlay (in our example scenario, *iotugr.es*), any joining node must know at least one node which is currently a member of the overlay. This is a common issue in P2P networks and is referred to as the bootstrapping process.

In RELOAD, the bootstrapping nodes (*i.e.*, the nodes used to form the first connection to the overlay) must have public IP addresses so that new nodes can reach them. Once a peer has connected to one or more bootstrap nodes, it can form connections using existing RELOAD operations.

In our system, the bootstrap node also acts as the certificate authority (located at *iotugr.es/bootstrap*), providing the joining node a Node-ID (an integer, e.g. 1234567) and a certificate associated to that Node-ID.

Once a peer has connected to the overlay for the first time, it can cache a list of public IP addresses the peer has successfully reached, and use these addresses as

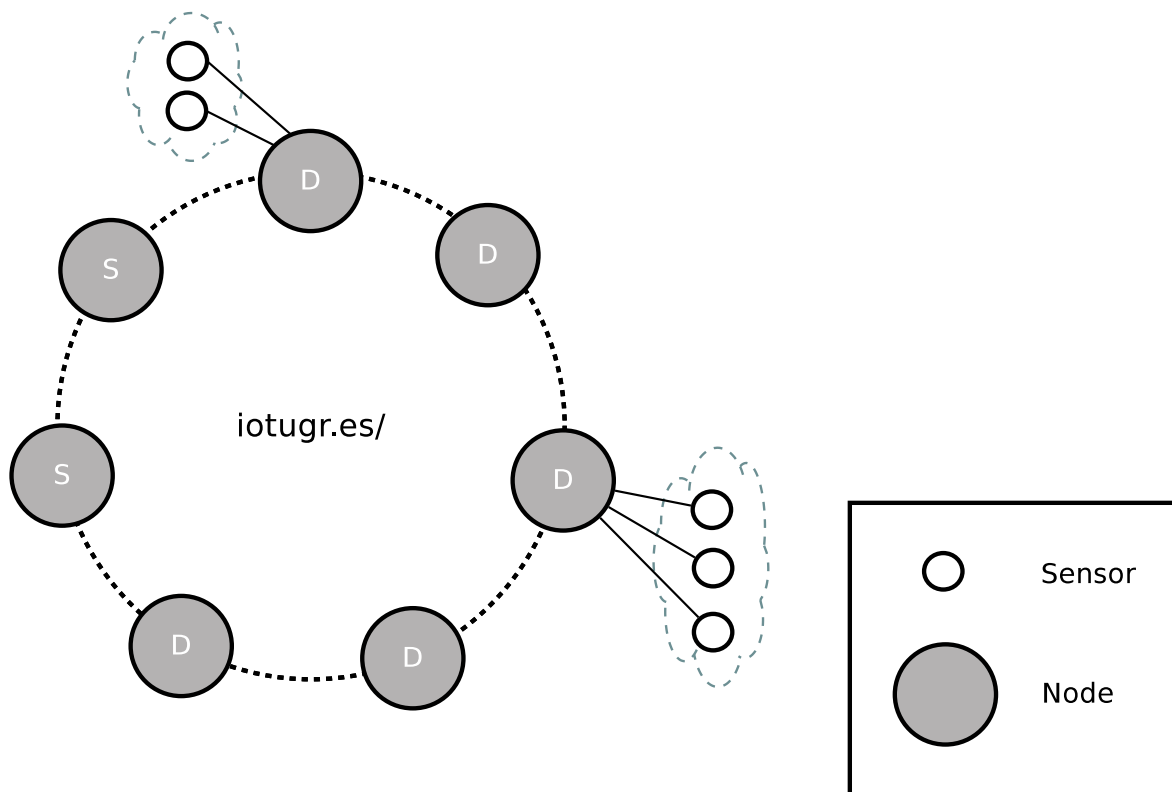


Figure 5.2: Example of overlay prior to any DataRegister.

future bootstrap nodes.

After joining is completed (see Figure 5.2), the joining peer is a full member of the overlay and can process Store/Fetch requests.

5.5.2 Registering entities

In this section we explain the procedures for registering the three different types of DataRegistration: TopicPublisher, TopicSubscriber, and DataSpaceParticipant.

5.5.2.1 Registering a new publisher using TopicPublisher

Continuing with our example, when the user *jmlvega* installs a IoT-enabled alarm clock, the clock registers itself into the *tevent* topic as a publisher (in our example, this topic contains time-driven events, such as calendar alerts or alarm clock events). In order to register, the alarm clock performs a RELOAD Store using the following URI:

```
dcps://iotugr.es/jmlvega/home/tevent/.pub/
```

Since Chord uses SHA-1 hashes for addressing resources, the node calculates the SHA-1 of the URI and then performs a store of the Kind "TOPIC_PUBLISHER_REG" with the following data:

```
Resource-ID:      53b573958039726dec72d3ce0e06367f569be4db
DICTIONARY-KEY:  1234567
VALUE:           {../alarmclock}
```

After performing a successful store, this information is available in the overlay and can be fetched later by any node. In order to provide data privacy, stored data-content can be encrypted using a public key. In this way only authorized nodes (the ones with the proper private key) will be able to read the stored data later.

5.5.2.2 Registering a new subscriber using TopicSubscriber

Locked doors that are accessible to the user *jmlvega* (his own house's doors, and certain doors at his workplace) can use TopicSubscriber for registering their discovery information. In this way, the user can connect to a particular lock and then publish the proper open/close signal. Since door locks are usually constrained devices, they are typically accessed through a proxy. In our example, garage and front doors of the user's house register through the *domopc* node, which acts as a proxy. In order to register the locks, *domopc* performs a RELOAD store using the following URI:

```
dcps://iotugr.es/jmlvega/lock/.sub/
```

Once the SHA-1 has been calculated, the node performs an store of the Kind "TOPIC_SUBSCRIBER_REG" with the following data:

```
Resource-ID:      d15138ddb356e8e6726c6985cea423c9cae7a28
DICTIONARY-KEY:  0000000
VALUE:           {../..../home/domopc/garaged ;
                 ../..../home/domopc/frontd  }
```

As in the previous case, this information will be available for any member of the overlay, and therefore it should be encrypted in order to provide data confidentiality.

The fact of registering both publishers and subscribers (instead of registering only publishers) enable nodes to discover matching nodes as soon as they join the network, without relying in periodic fetches from subscribers. Additionally, it eases the balancing of load and requests among the overlay.

5.5.2.3 Registering a new data-participant using DataSpaceParticipant

There are situations where nodes need access to a significant number of topics or even to a complete data-space. For example, a domotic house central computer should be able to discover all the sensors of the house, even if those sensors publish information through different topics. For these cases, we have designed DataSpaceParticipant Registration.

Continuing with our example, the computer at the domotic house could register itself to the data-space *jmlvega/home* using the following URI:

```
dcps://iotugr.es/jmlvega/home/.par/
```

Once the SHA-1 has been generated, the node performs an store of the Kind "DATA_SPACE_REG" with the following data:

```
Resource-ID:      4bffc1a6f31a40b054584f27464f84ba2ac32316
DICTIONARY-KEY:  0000000
VALUE:           {../domopc                               [DP]}
```

In this store, the node has registered only one URI as a data-participant (DP). Registering the type of the DCPS entity (in this case, data-participant) eases the processing of rendezvous information. Thus, the fetching nodes can filter dictionary entries according to their interests.

Our system also supports nodes that host several DCPS entities. Consider a mobile device that simultaneously maintains a calendar, a domotic house control panel, and an application for registering alerts generated by a domotic house. This device will perform the following store for registering three publishers (in this case, it uses the same Resource-ID, but a different Node-ID):

```
Resource-ID:      4bffc1a6f31a40b054584f27464f84ba2ac32316
DICTIONARY-KEY:  5996172
VALUE:           {../mobile/calendar                       [PB];
                  ../mobile/domocontrol                   [PB];
                  ../mobile/domoalerts                    [PS]}
```

Figure 5.3 depicts the overlay after performing the DataRegistrations described in our example. In the following sections we study how nodes can retrieve this information.

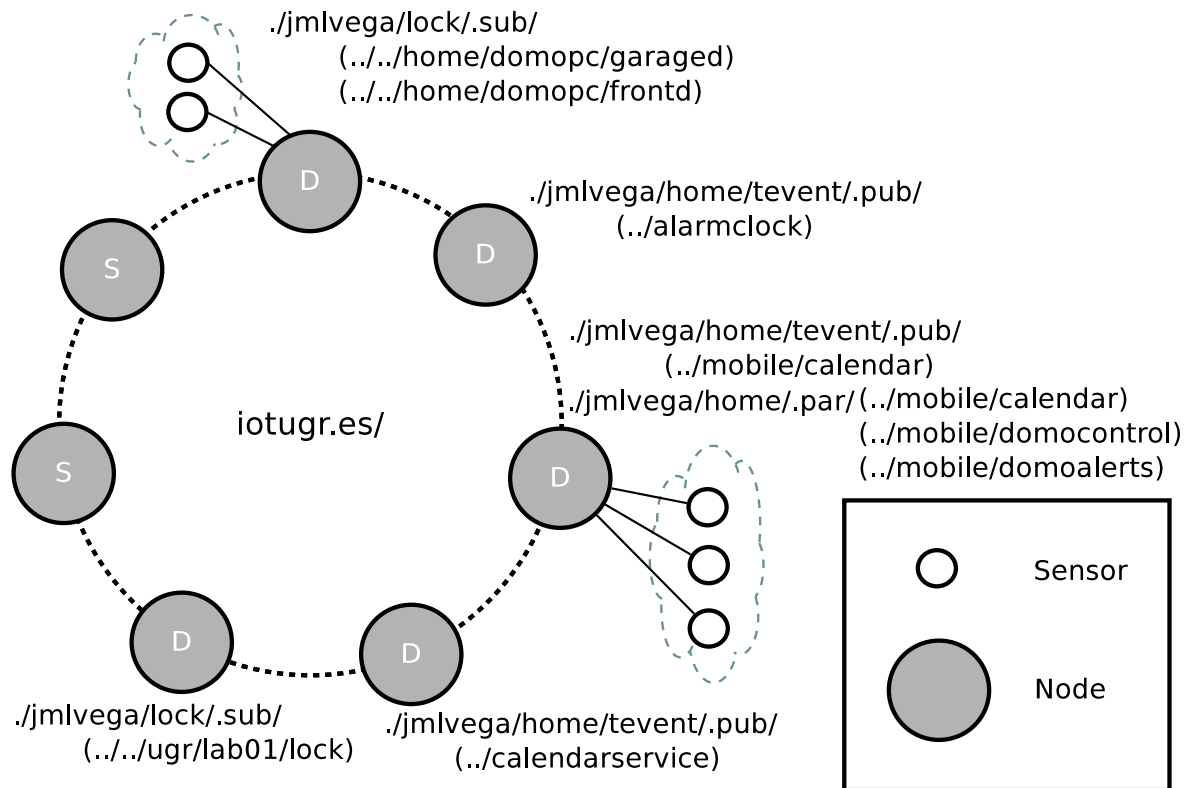


Figure 5.3: Example of overlay after several `DataRegister`.

5.5.3 Entity Discovery

In this section we explain the procedures for discovering nodes in the overlay using the three available `DataRegistration` types: `TopicPublisher`, `TopicSubscriber`, and `DataSpaceParticipant`.

5.5.3.1 Publisher discovery

Continuing with our example, a calendar application needs to retrieve planned events from a user data-space. After joining the overlay, the node that hosts the application calculates the SHA-1 for the following URI:

```
dcps://iotugr.es/jmlvega/home/tevent/.pub/
```

Once the SHA-1 is obtained, the node sends a fetch using the SHA-1 as the `Resource-ID`, and `"TOPIC_PUBLISHER_REG"` as the `Kind-ID`. This operation returns the following entries:

```

Resource-ID:          53b573958039726dec72d3ce0e06367f569be4db
1. DICTIONARY-KEY:   1234567
   VALUE:             {../alarmclock}

2. DICTIONARY-KEY:   5996172
   VALUE:             {../mobile/calendar}

3. DICTIONARY-KEY:   6352145
   VALUE:             {../calendarservice}

```

The fetch has returned three results. The first one is the one in the example at Section 5.5.2.1. The other two entries correspond to calendar services that had registered its presence information later. With this information, the application is able to access those publishers through the overlay and then request for the data-content.

5.5.3.2 Subscriber discovery

After joining the overlay, a user application for managing multiple locked doors performs a fetch to the Resource-ID associated to the URI `dcps://iotugr.es/jmlvega/lock/.sub/`. Since the application is looking for subscribers, it uses "TOPIC.SUBSCRIBER_REG" as Kind-ID. As a result, it obtains the discovery information of subscribers associated to that data-space and topic:

```

Resource-ID:          d15138ddb356e8e6726c6985cea423c9cae7a28

1. DICTIONARY-KEY:   0000000
   VALUE:             {../../../../home/domopc/garaged ;
                      ../../home/domopc/frontd  }

2. DICTIONARY-KEY:   9456721
   VALUE:             {../../../../ugr/lab01/lock}

```

From this point, the application is able to connect to those nodes using RELOAD AppAttach method, and then the user can select what door lock he wants to activate.

5.5.3.3 Data-participant discovery

Continuing with the example in Section 5.5.2.3, if the computer at the domotic house performs a fetch to the Resource-ID associated to the URI `dcps://iotugr.`

es/jmlvega/home/.par/ using "DATA_SPACE_REG" as Kind-ID, the obtained result will be:

```
Resource-ID:          4bffc1a6f31a40b054584f27464f84ba2ac32316

1. DICTIONARY-KEY:   0000000
   VALUE:             {../domopc          [DP]}

2. DICTIONARY-KEY:   5996172
   VALUE:             {../mobile/calendar  [PB];
                      ../mobile/domocontrol [PB];
                      ../mobile/domoalerts [PS]}
```

Where the first entry corresponds to its own discovery information. Using these results, the computer is now able to connect to those nodes and, therefore, to exchange information with them.

5.5.4 Data transfer

Once a node has obtained a list with Node-IDs and URIs associated to matching nodes, it can establish a connection to those nodes. This connection is established using RELOAD AppAttach (see Section 2.4.3).

Our system has been designed to be generic. In this sense, nodes can perform data transfer using any publish-subscribe based protocol. For example, applications can use CoAP [97] for exchanging data-content over an existing RELOAD AppAttach connection. CoAP is an emerging protocol suited for working with low-power devices (the ones typically used in IoT deployments). Consequently, a CoAP-based solution allows for integrating these constrained devices to the overlay.

Another possible alternative is to use DDS-RTPS [69]. DDS-RTPS is part of the DDS standard family [67] and it is defined for transferring data in DCPS systems. DDS-RTPS solution is adequate for medium to high complexity distributed systems that require advanced features as data reliability, QoS negotiation, or data persistence.

In the following sections we describe the process of data transferring using these two protocols.

5.5.4.1 Data transfer using CoAP

Once a node that hosts a subscriber (referred to as I-Node) receives the rendezvous information of a node that hosts a publisher (referred to as T-Node), I-Node sends an AppAttach request to T-Node. Upon AppAttach reception, T-Node generates an AppAttach response that establishes a connection between I-Node and T-Node.

Using this connection, the I-Node sends a subscription request to the T-Node. This request follows the procedure described in [28]. In particular, I-Node sends a GET request (using the open connection) to the URI associated to a particular DCPS-Entity-Name (e.g., `coap://iotugr.es/jmlvega/home/tevent/mobile/calendar`). This URI is the one obtained during the rendezvous process, after replacing `”dcps://”` with `”coap://”`. From this point, the publisher *mobile/calendar* delivers its updates to the monitoring node using CoAP notifications over the active connection.

If the initiator of the dialog is a publisher, the procedure is almost the same as the described for a subscriber, but an additional step is necessary: the node that contains the publisher must notify subscriber’s node about the existence of an active publication.

5.5.4.2 Data transfer using DDS-RTPS

Once a node that hosts a DDS participant (referred to as I-Node) receives the rendezvous information of a node that hosts another DDS participant (referred to as T-Node), I-Node sends an AppAttach request to T-Node. Upon AppAttach reception, T-Node generates an AppAttach response that establishes a connection between I-Node and T-Node.

Once rendezvous is finished, DDS implementations can perform their own PDP and EDP procedure (for more information about DDS/DDS-RTPS discovery, please refer to Section 2.2.2). After completing the DDS-RTPS discovery process, nodes are able to push updates from DWs to matching DRs. This data flow continues as long as the AppAttach connection remains active and both matching DW and DR are active.

5.6 Prototype and experimental setup

To validate our proposal, we have implemented a prototype. It is an extension to the one used in [58], [59], and [57], which is a Java Standard Edition (J2SE) implementation of RELOAD owned by ERICSSON-RESEARCH.

Our prototype implements the three different Registrations proposed in Section 5.4.5. In this regard, it allows overlay peers to discover three different kinds of DCPS entities (data-participants, publishers, and subscribers). Regarding content addressing, we have used the URI format proposed in Section 5.4.6.

In order to perform stores and fetches, the URIs are encoded using SHA-1. The prototype implements a CoAP-based procedure for transferring data between peers.

We have conducted a series of experiments to validate our prototype. In the next subsection, we detail the experimental setup.

5.6.1 Experimental setup

Our prototype includes the same code base as the RELOAD/P2P-SIP prototype used by ERICSSON-RESEARCH in [54], [55], and [56] to run experiments in PlanetLab [64]. For the experiments reported in this chapter we ran multiple instances of our prototype over a simulated overlay. We decided to use a simulated overlay to be able to experiment with large-scale overlays (up to 10000 nodes). The validity of the measures of the used simulator has been already demonstrated in [57], which includes a comparison between simulation and real PlanetLab deployment results.

Our simulator uses a topology generator that assigns peers randomly to 206 different locations around the world. These locations correspond to PlanetLab sites. The pairwise delays between peers were set based on real pairwise delays that we measured between nodes at these PlanetLab sites.

We have chosen Chord as the DHT algorithm for creating and organizing the overlay. The reason to use Chord is because RELOAD specifies it as the mandatory-to-implement DHT. The size of Chord's successor list and finger table were set to $\log_2 N$, while the size of predecessors list were set to $0.5 * \log_2 N$. Where N is the maximum network size for each experiment (as we will see later, it ranges from 500 to a maximum of 10000 nodes).

We have assumed an unreliable transport protocol for peer communications.

We have also assumed a 90% of peers were located behind a NAT that uses endpoint-independent mapping and filtering behavior. In this sense, our simulations consider the time of ICE negotiation for performing NAT traversal when necessary.

In our system, data reliability and system robustness is a main concern, as discovery information loses prevent nodes for being discovered at Discovery-and-Data layer.

In order to provide robustness against node failures and ungraceful disconnec-

tions, we configured our prototype for replicating among multiple peers the information stored in the overlay. We have chosen a replication factor equal to 3 for the stored DCPS registers (*i.e.*, information is replicated in three additional nodes). We have assumed a 10% of peer leaves are crashes.

In Section 5.4.3 we defined two node roles: S-NODES and D-NODES. In order to test the worst-case-scenario, we have assumed all the nodes in the overlay are D-NODES (*i.e.*, every node stores its discovery information in the overlay, and therefore generates the associated additional traffic).

In the experiments, each D-NODE registers one DCPS entity to the overlay. The kind for this entity is generated randomly. Specifically, 20% of the nodes register a DCPS data-participant, 40% of the nodes register a DCPS publisher, and 40% of the nodes register a DCPS subscriber.

The simulator generates the URI associated to each register randomly from a pool of URIs. Since the URI format is different for each new RELOAD Kind, the simulator has three different URI pools: one for data-participants, one for publishers, and one for subscribers.

We have tested both the rendezvous and AppAttach procedures. In this sense, each node performs a lookup for a randomly generated URI and starts an AppAttach with all the obtained entries (*i.e.*, all the entities of the same DCPS-Entity-Group). This is also a worst-case scenario, as in the typical use-case the node will not necessarily open a connection with all the discovered nodes, but a subset of them. The simulator obtains lookup URIs from the same three pools used during the store; however, in this case Publishers use the Subscriber URI pool and *vice versa*.

In order to obtain comparable experimental data from different network sizes, the average number of entries per Resource-ID should be the same for all the experiments (if not, the different size of lookup responses will impact on the results). Our simulator achieves this behavior by adjusting the URI pool sizes proportionally to the maximum network size.

In our experiments, we have assumed an average of 20 entries per Resource-ID. This means a node will connect an average of 20 nodes after a successful lookup result (*i.e.*, the nodes that share a particular segment of the DCPS data-space).

We have conducted two different types of experiments for testing the system scalability and robustness. In the first set of experiments, we study how the number of nodes in the overlay (from 500 to 10000) impacts the system performance (*i.e.*, the average latency for stores, lookups, and inter-peer connection establishment).

In the second set of experiments, we evaluate how the churn rate (modeled as the rate at which nodes join and leave the overlay) impacts the reliability of the system.

5.7 Experimental results

The first set of experiments evaluates the scalability of the system. In particular, we measure the incurred latencies for storing and obtaining the proposed RELOAD Kinds in different network sizes (specifically 500, 1000, 5000, and 10000 nodes).

We have also measured the time necessary for discovering a particular DCPS-Entity-Group and establishing a connection with a node associated to that group.

In order to measure these latencies, the simulations have two phases: overlay creation and measurement.

During the overlay creation phase, the simulator creates an overlay with approximately N nodes, where N is the tested network size. In this phase, the simulator creates new nodes, whose locations are randomly chosen from our PlanetLab database, and joins them to the overlay.

Once a particular node has joined to the overlay, it performs a store to one of the three new proposed RELOAD Kinds (chosen randomly as we describe in the previous section).

Once the ring is created, the measurement phase starts: during this phase new nodes join the network, perform a store, obtain the discovery information for a randomly chosen DCPS-Entity-Group, and open a connection with all the obtained nodes.

The join rate during this phase has been modeled as a Poisson process with an average of 0.5 joins per second. The simulated time for this phase was 3600 seconds, and each experiment has been repeated three times (for a total simulation time of 10800 seconds per experiment).

Figure 5.4 depicts the average and 95% percentile latency associated to the store and lookup processes of the new proposed RELOAD Kinds, for multiple network sizes. The results show that the system scales well in terms of latency as the number of nodes in the network increases. The average and 95% percentile latency for storing the discovery information of a DCPS entity follows a logarithmic growth as a function of network size.

In the same way, the average and 95% percentile latency for retrieving the discovery information of all the nodes associated to a particular DCPS-Entity-Group also follows the same trend.

In addition to the store and lookup latency, we have measured the total time from the request of a particular DCPS-Entity-Group (*i.e.*, after starting the rendezvous) until the creation of a AppAttach connection (using ICE if necessary) and the arrival of the first CoAP update. Figure 5.5 shows the average delays for dif-

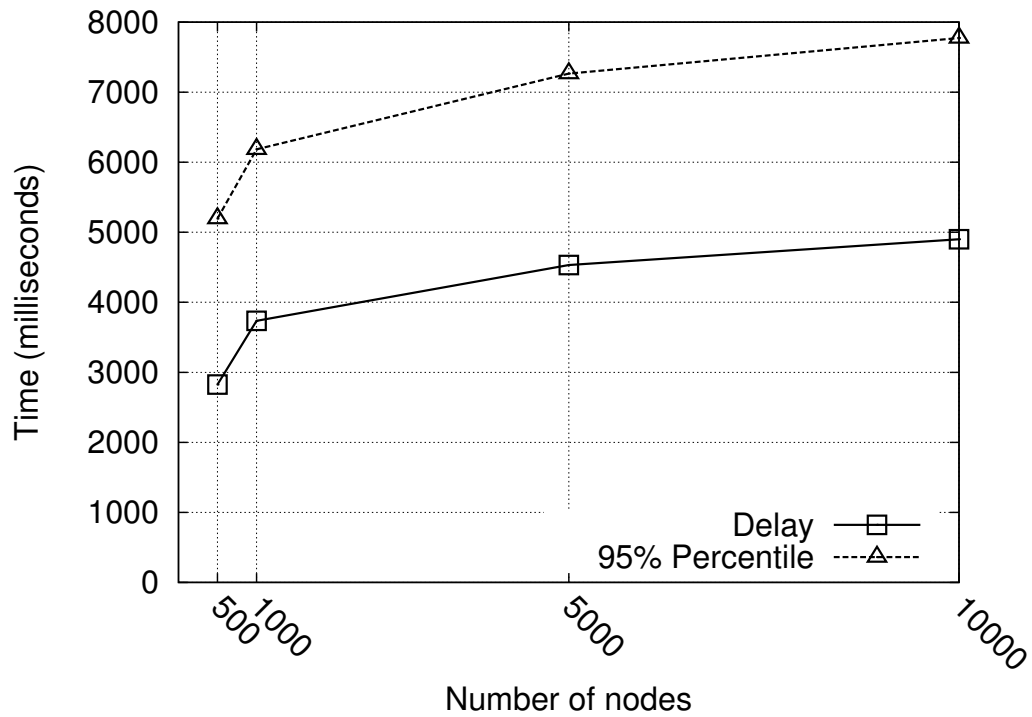
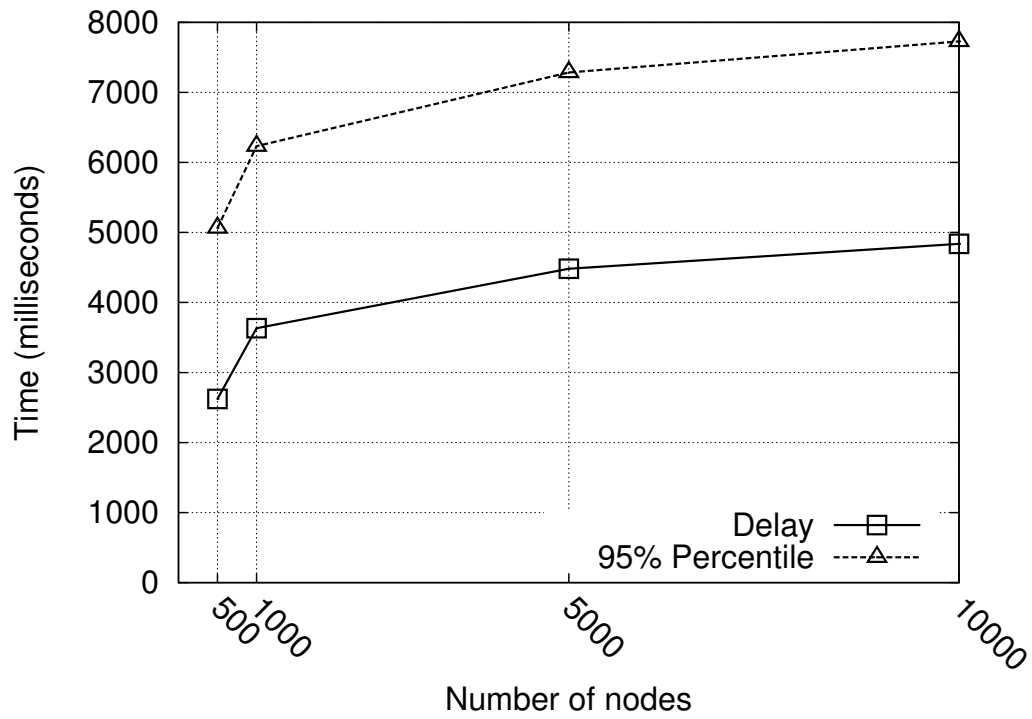


Figure 5.4: A) Average latency for storing DCPS entities discovery information. B) Average latency for obtaining all the discovery information of a particular DCPS-Entity-Group.

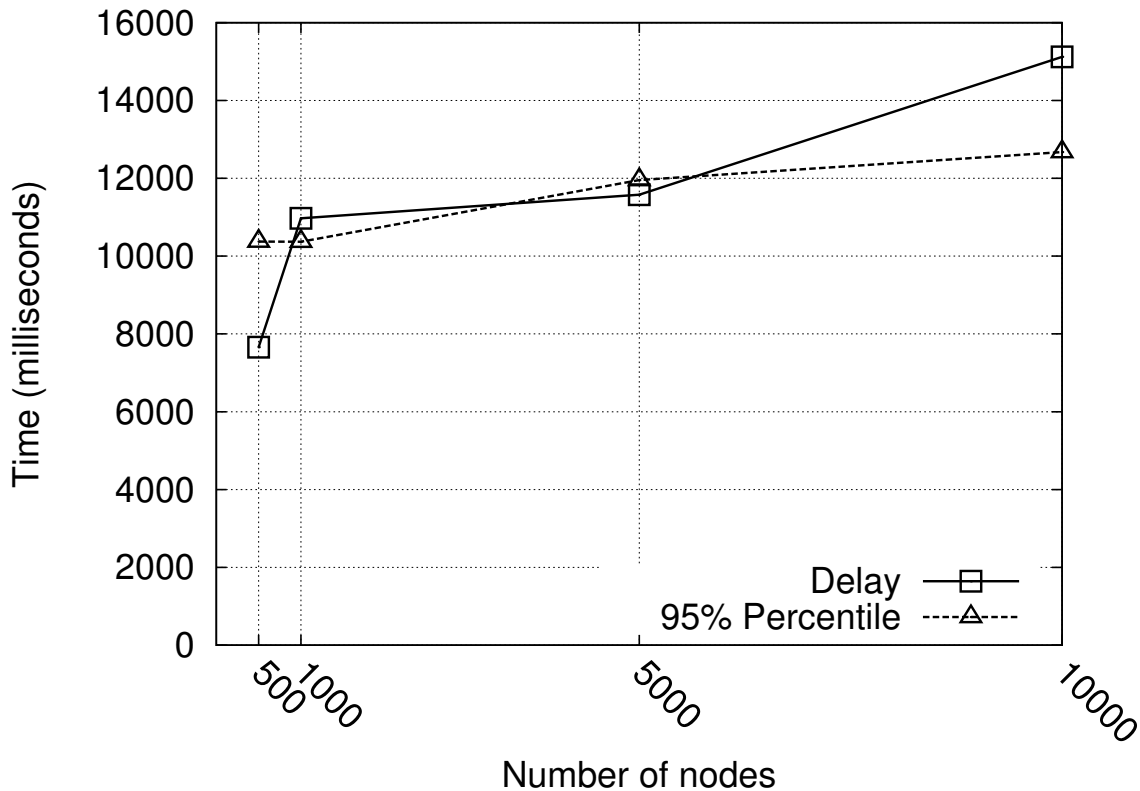


Figure 5.5: Average latency for obtaining the discovery information for a particular DCPS-Entity-Group and start exchanging data with a node of that group.

ferent network sizes. Although the average presents certain deviation (due to a reduced number of measures with a very high latency), the 95% percentile grows logarithmically as a function of the network size. From the obtained results we can conclude that the average time for discovering and connecting to a set of nodes of interest scales well as the total size of the network increases.

In the second set of experiments, we have studied the robustness of our system, evaluating the impact of the churn rate on reliability. We have measured the percentage of lookup fails when the overlay is suffering a particular churn rate, modeled as a combination of join rate and leave rate. Specifically, we have simulate rates of 0.1, 0.2, 1, and 2 join/leaves per second.

As in the case of the scalability experiments, each simulation has two phases: overlay creation and measurement.

During the first phase, the simulator creates an overlay of approximately 5000 nodes. After the overlay creation, the simulator simulates a particular churn rate (*i.e.*, a particular join and leave rate). After each new join, the joined node performs a

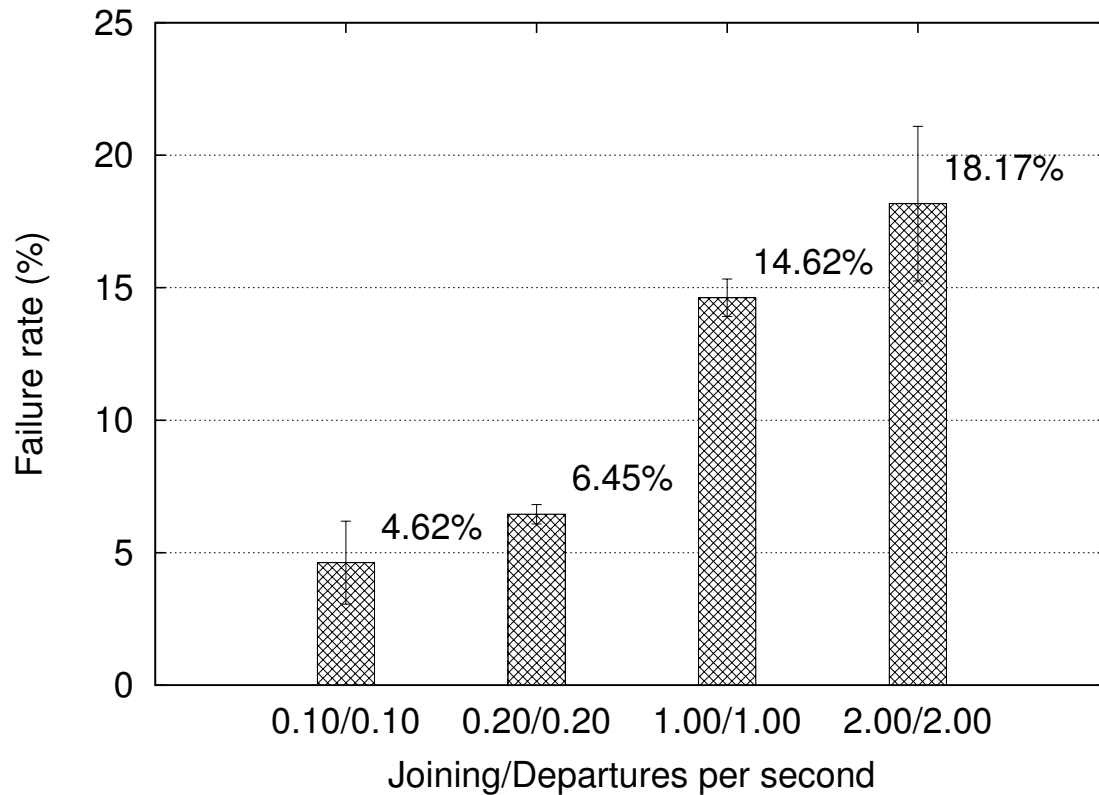


Figure 5.6: Percentages of lookup failures per different churn levels.

lookup. The simulator stores the result of this lookup and calculates the percentage of lookup fails once the simulation is over. During the simulation, as we stated in the previous section, a 10% of leaves are node crashes.

In the Figure 5.6 we include the different lookup failures percentages for the simulated churn rates. As expected the system is robust, and the percentage of failures for even high levels of churn (we have stressed the system with rates up to 2 joins and 2 leaves per second, which makes the topology of the system very unstable) remains below 18.17% of failures.

5.8 Conclusions

In this chapter, we have proposed a solution for performing content discovery and data transfer on large-scale DCPS systems.

Our solution is based on the RELOAD framework, one of the latest standards of the IETF. In particular, we have defined three new RELOAD Kinds that ex-

tend its functionality. These new Kinds cover three different DCPS entities: data-participants, publishers, and subscribers.

Our system provides rendezvous function for locating sensors, actuators, and applications that share a common data-space or topic of interest.

Once the rendezvous is complete, nodes use existing RELOAD features for establishing a connection (our solution provides NAT traversal using ICE). User applications can use this connection for transferring data using a publish-subscribe approach.

Our proposal is extensible, and any publish-subscribe application can benefit from it, for instance, those based on CoAP or on DDS.

In order to validate our proposal –and evaluate its scalability and robustness–, we have conducted a series of experiments over a simulated network of 500 to 10000 peers. The topology of this network is based on real-world data, specifically from the PlanetLab network.

The obtained results show that our proposal is scalable, as the average delay for the main operations of our system follows a logarithmic growth as a function of the network size.

Regarding robustness, we have stressed a 5000-sized network with different churn rates. The obtained experimental results show that the percentage of lookup failures remains low even for very high churn levels. In this sense, for a level of churn equivalent to 2 joins and 2 leaves per second (*i.e.*, the 25% of the nodes change every 10 minutes), the percentage of lookup failures is lower than 18.17%.

Related Work

IN this chapter we select some of the works presented in Section 1.4 that are specially related to our work, and we compare their strengths and weaknesses in relation to our work.

In particular, we review a number of publications that address the system federation and domain bridging problems, with special attention to the ones focused on DDS. We also study research works focused on the integration of DDS and SIP and compare these works to our DCSIP-based solution. We review several research works that study the deployment of publish-subscribe scenarios using P2P-based architectures. We also study a RELOAD usage that is very close to the one proposed in Chapter 5. Finally, we compare the NDN model with the solution proposed in Chapter 4.

6.1 System federation and domain bridging in DDS

In this Thesis, we have addressed the problem of system federation and domain bridging for DDS systems. As we studied in Section 1.4.2, In the literature we can find other research works that also address this problem. In this section we review those works, and compare with the proposed DDS-IS.

First, we studied the Apache QPiD project [4], which is an implementation of the AMQP [3] that includes a broker-based federation mechanism. This federation mechanism increases the scalability of AMQP-based messaging systems by establishing dedicated routes between Apache QPiD brokers. These routes are similar

to DDS-IS routes, as they establish virtual channels between applications that exchange information. However, Apache QPiD is not data-centric, and consequently does not support content-based filtering nor data transformation as the DDS-IS does.

Also related to AMQP, we studied [60], which provides a systematic method for configuring broker federation in AMQP-based systems and they conduct a set of experiments for evaluating their method. In one of their experiments, they evaluate a single broker that sends messages to a variable number of consumers. Although the results are not directly comparable to the obtained for the DDS-IS (they use a different testbed), we can evaluate the degradation of the performance when scaling from 1 to N consumers. In particular, for the case of eight consumers and data-sizes ranging 64B to 512B (the most common in DDS applications), they experimented a degradation of the performance of the 90%, while in our experiments this degradation is only of 25%. In addition, they do not support data-centric features like content-based filtering or data transformation.

In Section 1.4.2 we also studied [108], an scalable publish-subscribe architecture for Mobile ad hoc network (MANET)s. In this work the authors deal only with scalability problem and, although they perform content-based routing, they leave open issues such as data transformation, which is covered by the DDS-IS.

Related to DDS, we studied REVENGE [16] [17], which is a middleware for distributing breaking news among heterogeneous clients. Although this work allows nodes located in different domains to exchange information, the interconnected nodes have to use a specific topic/data-type. Our proposal does not have this limitation, as it takes advantage of the DDS-XTypes specification for exchanging data-types unknown at compilation time.

Finally, our work is complementary to [77], which proposes an architecture to interconnect DDS data-spaces with ESB, because DDS-IS provides mechanisms for transforming and aggregating the exchanged information beyond the adaptation between ESB and DDS that this work provides.

The work presented in this Thesis is complementary to these works in the sense that it simultaneously solves three issues that currently exist in DDS: system scalability, data transformation between data-spaces, and QoS adaptation between remote entities. To the authors knowledge, it is the first work (compliant with OMG standard) that solves such issues by proposing a topic bridging architecture for DDS.

6.2 DDS and SIP

As we mentioned in Section 1.4.4, in [51] we proposed the concept of DDS session. This Thesis extends this work by defining an architecture that supports DDS sessions and by proposing unique identifiers for DCPS entities and content. In addition, this Thesis proposes a P2P-based architecture for solving DDS discovery, instead of the client-server architecture [51] is based on.

We also studied [26], which proposes a system for integrating SIP and DDS. This solution is limited by the client-server architecture SIP is bound to. In addition, SIP is not a protocol designed for data-centric publish-subscribe environments, and therefore is not so suited for session signaling in DDS environments as the proposed DCSIP is.

6.3 RELOAD and publish-subscribe scalability

A number of papers have aimed to demonstrate the feasibility of RELOAD as a stable and reliable P2P architecture. For instance, RELOAD performance in wireless environments has been studied in [58]. In [56], a study of the performance of RELOAD on ICE-based environment for NAT traversal is done. Another contribution is [28], which addresses the problem of resource discovery and notification in CoAP environments.

Scalability in publish-subscribe systems is an important issue. Discovery and dissemination of publications in large deployments are challenging problems, specially due to the great amount of resources required to maintain a database of all the publications and subscriptions. The paper [92] addresses the problem of scalable discovery in DDS environments by applying Bloom filters. However, this work only focuses on the discovery problem itself, not addressing other issues such as maintaining an overlay to provide fault-tolerance, or NAT-traversal.

To provide fault-tolerance, the authors of REVENGE [16] propose a self-organizing P2P substrate built atop of DDS to support DCPS communications. In REVENGE, instead of maintaining direct communication between publishers and subscribers, an overlay is utilized for routing the publications, thus alleviating the data sources from the responsibility of disseminating publication updates to each interested subscriber. REVENGE relies in the deployment of relay nodes that manage different data-spaces. Our proposal supports this behavior, and it also allows for distributing the same data-space across multiple peers in the overlay, regardless of their location.

Finally, in Section 1.4.3 we studied [2], which proposes a publish-subscribe pro-

protocol built atop of RELOAD. This proposal relies on tunneling mechanisms for information exchange. Our work, instead, relies on RELOAD AppAttach mechanism: once the publications are discovered and matched to interested subscriptions, the communication phase is done either by CoAP or DDS, thus ensuring all the features of these technologies (such as minimal resource usage or QoS requirements). Moreover, this work does not cover, as our work does, the discovery of specific DDS entities, such as participants.

6.4 CoAP usage for RELOAD

We proposed a new CoAP usage for RELOAD in [45]. This usage provides the functionality to federate Wireless Sensor Networks (WSN) in a P2P fashion. It also provides a rendezvous service for CoAP Nodes, and supports the caching of sensor information. This usage takes advantage of the RELOAD AppAttach method to establish a direct connection between nodes through which CoAP messages are exchanged.

The CoAP usage for RELOAD (proposed in [45]) and the DCPS usage for RELOAD (proposed in this thesis) use the same data model (a dictionary) and URI-building mechanism. However, the CoAP usage for RELOAD does not support the three DCPS registrations that the DCPS usage supports (TopicPublisher, TopicSubscriber, and DataSpaceParticipant), and which are necessary for supporting DCPS environments.

6.5 Named Data Networking (NDN)

The current Internet design, mostly based on request-response exchanges, is now opening to completely different paradigms. In particular, the authors of the seminal work [41] state that one of the issues of the current Internet design is the location-dependence. In this sense, they consider that mapping content to host locations complicates configuration as well as implementation of network services.

To address this and other issues associated to the current Internet design, they proposed the NDN model [109]. In contrast to the 'conversational' nature of IP, in which IP datagrams can only name communication endpoints (the IP destination and source addresses), NDN proposes to generalize the Internet architecture by removing this restriction. In this regard, NDN networks route datagrams using names, which are hierarchically structured identifiers.

NDN names are compatible with the concept of scope and scope-path we pro-

posed in Chapter 4. In this sense, scope and scope-path organize content hierarchically by using names associated to each node. Consequently, we believe that the proposed DCSIP protocol will allow to use DDS over *NDN* networks in the future.

6.6 Conclusions

In this chapter we have presented some related works to our research. In particular, we have studied some works that address the problem of system federation on publish-subscribe systems, we have reviewed some research works focused on the integration of SIP and DDS, we have studied a set of works focused on deploying publish-subscribe systems over P2P architectures, and we have compared our solution with the *NDN* model.

We can conclude that our work is complementary to the ones found in the literature, as it provides a complete solution for several currently unresolved DDS issues. In particular, we have proposed an efficient platform for data-centric routing and data transformation (the DDS-IS) that is compliant with existing DDS specifications. Although we can find several works that cover one or two of these features, there are no other works that simultaneously solve all these problems while maintaining standards compliance.

Regarding signaling on DCPS systems, there are no other works in the literature that propose a DCPS-based signaling protocol. Instead, some works have focused on integrating SIP and DDS. However, as we studied in Chapter 4, SIP does not suit well to the specific requirements of DCPS-based environments.

We can find some works that propose P2P-based architectures for supporting publish-subscribe systems. However, none of these works addresses the unique requirements of DCPS-based systems, such as DDS. In this regard, we have proposed a RELOAD-based solution for solving DCPS entity and content discovery that is scalable and robust.

Finally, we have studied how our proposal is well aligned with the *NDN* model, that addresses content using names that describe data, instead of names describing data location.

Conclusions and future work

THIS chapter summarizes the main conclusions of our work and discusses some open issues for future research.

7.1 Conclusions

Publish-Subscribe communications have been gaining increasing popularity. Notably recent efforts to define a future publish-subscribe-based Internet are [20] [22] [82]. DDS is one of the more popular standards for providing publish-subscribe communications and appears well suited to the needs of the emerging "Internet of Things." While powerful, DDS is a fairly new standard and leaves undefined important issues for global Internet deployment such as universal DCPS entity and data-content naming, end-to-end session establishment, data-space interconnection, data transformation, QoS adaptation, and scalability to very large deployments. This Thesis addresses many of these shortcomings.

In Chapter 3 we proposed the DDS-IS, a solution for the interconnection of DDS data-spaces. In addition to the interconnection problem, the DDS-IS also addresses three still unresolved issues of DDS: data transformation between data-spaces, system scalability, and QoS adaptation between remote entities. Moreover, and following the data-centric philosophy of DDS, the proposed service is able to perform content-based filtering of the exchanged data.

The design of the DDS-IS features an efficient data transforming capability, which transforms data as it flows between applications. As a result, multiple ver-

sions of the same application, or even applications of different vendors, can now seamlessly interoperate, without change, despite differences in data structures and interface definitions. DDS-IS also supports content-based filtering, which can be used for avoiding sensitive information of a given data-space to be exposed to other data-spaces.

One of the key benefits of our solution is its standards compliance. The DDS-IS is fully compliant with the latest specifications of the DDS standard family. As a result, our design is not implementation-dependent, and hence it is applicable to any existing or future DDS implementation. Namely, the DDS-IS benefits from the DDS-XTypes [71], a new OMG specification for accessing to topic types discovered at execution time.

We implemented a DDS-IS prototype that demonstrates the advantages of our proposal. Based on this prototype, we reported experimental results to evaluate the performance impact of the proposed service. The reported results show that the DDS-IS has a very low impact on DDS performance, in terms of overhead latency and achievable throughput. We also demonstrated that the DDS-IS can improve the scalability of DDS systems by reducing the network traffic load between remotely interconnected data-spaces. Finally, we conducted a study of the impact of DDS-IS in DDS QoS policies provision. From this study, we conclude that QoS enforcement is applicable only in a hop-by-hop basis (*i.e.*, among entities within a particular domain) for most of DDS QoS policies. There are, however, two exceptions: LIVELINESS and LIFESPAN. These two DDS QoS policies can be enforced on a end-to-end basis (*i.e.*, across multiple interconnected domains).

The DDS-IS does not address the problems of universal entity identification and universal data-content identification, end-to-end session establishment, and DDS discovery scalability in large deployments. To solve these problems, in Chapter 4 we defined a mechanism for universal identification of DCPS entities and data-content and we proposed the design and rationale of a protocol for session signaling in DCPS environments. This protocol provides DCPS entity and data-content discovery in medium-scale environments. It also allows the creation of sessions between nodes, which eases the creation of DCPS routes and the enforcement of QoS at end-to-end level.

The protocol has not been implemented at the time of writing. However, one of the main implementors of DDS, RTI Inc. [89], has already shown its interest in DCSIP implementation, which envisages a very promising future for the proposal.

In spite of its benefits, the proposed solution is not suitable for very large DCPS environments, as the presence of a unique rendezvous server may not scale for very high-scale deployments (deployments with thousands of nodes). To address this issue, in Chapter 5 we proposed a scalable P2P solution for performing content discovery and data transfer.

This solution is based on the RELOAD framework, one of the latest standards of the IETF. In this sense, we proposed a RELOAD-based architecture for addressing the requirements of DCPS IoT scenarios. In addition, we define three new RELOAD Kinds that extend the current functionality of RELOAD. These new Kinds cover three different DCPS entities: data-participants, publishers, and subscribers.

Our system provides rendezvous function for locating sensors, actuators, and applications that share a common data-space or topic of interest. Once the rendezvous is complete, nodes use existing RELOAD features for establishing a connection. User applications can use this connection for transferring data using a publish-subscribe approach. In this regard, our solution provides the additional benefit of providing NAT traversal using ICE.

Our proposal is extensible, and any publish-subscribe application can benefit from it, for instance, those based on CoAP or on DDS.

In order to validate our proposal, and evaluate its scalability and robustness, we have conducted a series of experiments over a simulated network of 500 to 10000 peers. The topology of this network is based on real-world data. In particular, the simulated experimental setup was configured according to PlanetLab data.

The obtained results show that our proposal is scalable, as the average delay for the main operations of our system shows a logarithmic growth as a function of the network size. Regarding robustness, we have stressed a network with 5000 nodes using different churn rates. The results show that even for high level of churn (we have stressed the system with rates equivalent to change the 25% of the nodes in the network every 10 minutes) the percentage of lookup failures is lower than 18.17%.

To bring this section to a close, we will now review the initial objectives enumerated in Chapter 1, and we will now examine to what degree the presented work meets these objectives.

- **DDS data-space interconnection:**

- **Initial objective:** To design a data-space bridging service for DDS.

- * **Results:** In collaboration with RTI, we designed the DDS-IS and applied for a patent with number US 2011/0295923 A1 [18]. The preliminary design was also presented in the Real-Time OMG Workshop 2009 [76].

- **Initial objective:** To increase the extensibility and flexibility of DDS systems, easing DDS systems evolution.

- * **Results:** We took advantage of the recently standardized DDS-XTypes specification for performing content-based transformations in DDS-IS

- nodes. The proposed DDS-IS also supports the integration of DDS data-spaces with different QoS policies.
- **Initial objective:** To implement and evaluate the designed data-space bridging service for DDS.
 - * **Results:** We implemented a prototype of DDS-IS and conducted several performance and scalability experiments. We published the results of this evaluation in an indexed journal [53].
 - **Initial objective:** To study the impact of the proposed service in DDS QoS.
 - * **Results:** We conducted an analysis of the impact of DDS-IS in the existing DDS QoS policies. In particular, we have identified which of these policies can be enforced in a hop-by-hop or end-to-end basis using the current DDS specification.
- **Session signaling in DDS:**
 - **Initial objective:** To design an universal entity and data-content naming format for DCPS environments.
 - * **Results:** We proposed the FQDN for identifying domains, the FQTN for identifying topics, the DCPS-Entity-Group for identifying groups of entities of the same type (participant, publisher, or subscriber) associated to a particular FQDN or FQTN, and the DCPS-Entity-Name for identifying DCPS entities. This proposal is currently in peer review process in an indexed journal.
 - **Initial objective:** To design a protocol for DCPS entities discovery and session establishment in DDS.
 - * **Results:** In the OMG Real-Time Workshop 2010 we proposed using SIP for establishing DDS Sessions [51]. The initial work has evolved to two new different solutions: first, a data-centric session signaling protocol named DCSIP; and second, a DCPS usage for RELOAD. In this Thesis we have proposed the design of DCSIP, and we have designed and implemented a prototype of the DCPS usage for RELOAD. The proposed DCPS usage for RELOAD is currently in peer review process in an indexed journal.
 - **Initial objective:** To evaluate the scalability and robustness of the proposed solution.
 - * **Results:** We have conducted a set of experiments that demonstrate the scalability and robustness of the proposed DCPS usage for RELOAD. The results of these experiments are currently in peer review process in an indexed journal.

In addition to the initial objectives, we have accomplished the following additional objective:

- **CoAP usage for RELOAD:**

- **Objective:** To propose an extension to RELOAD that supports the connection to constrained nodes in a request-response fashion.

- * **Results:** In collaboration with ERICSSON, we have submitted a IETF draft named CoAP usage for RELOAD [45]. This draft defines an extension to RELOAD to provide a rendezvous service for CoAP nodes and to support the caching of sensor information. This draft was presented in the IETF 83 Meeting [1] and it is currently being discussed within the P2PSIP WG [36].

7.2 Future work

As future work, we are interested on researching in new possible extensions and applications of the proposed DDS-IS. For instance, we are interested on studying how the Bloom-based SDP proposed in [92] impacts on the scalability of DDS-IS-based deployments.

Cloud computing architectures have gained recently more and more attention in both industry and academia. In this regard, there are some projects that use DDS for exchanging Cloud monitoring information. However, these solutions are implementation dependent. We would like to use DDS-IS for connecting DDS-based applications in Cloud scenarios. Since DDS-IS allows applications to use types unknown at the time of application compilation, we believe DDS-IS can help to build generic and flexible DDS-based Cloud infrastructures.

In this Thesis we have proposed DCSIP, a protocol for session signaling in DDS deployments. In collaboration with RTI, we plan to implement a DCSIP API. After implementing the protocol API, we plan to integrate it with the DDS-IS. Once the integration between DDS-IS and DCSIP is completed, we are interested on integrating our DCSIP-based solution and the DCPS usage for RELOAD. This integration will allow to create autoconfigurable federated DDS domains that adapt to both medium and large scale DDS deployments.

Regarding the DCPS usage for RELOAD, we want to focus our research on introducing load balancing mechanisms for addressing scenarios with a large number of entities per DCPS-Entity-Group (*i.e.*, scenarios where the number of entries associated to the same Resource-ID is high). Finally, we have also plans to propose for its standardization the proposed DCPS usage for RELOAD.

Conclusiones y trabajo futuro

EN este último capítulo se resumen las principales conclusiones de nuestro trabajo, y se discuten algunas cuestiones abiertas que justifican la continuación de la línea de investigación abierta por esta Tesis.

7.1 Conclusiones

Las comunicaciones publicación-suscripción han ganado popularidad últimamente, como así queda demostrado por la existencia de proyectos para definir una Internet del futuro basada en publicación-suscripción [20] [22] [82].

DDS, uno de los estándares de publicación-suscripción más populares, parece adecuado para responder a las necesidades del llamado "Internet de las Cosas". Sin embargo, DDS es un estándar relativamente joven, por lo que aún quedan algunos problemas por resolver antes de su despliegue en Internet, a saber: la identificación universal de contenidos y entidades DCPS, el establecimiento de sesión extremo a extremo, la interconexión de espacios de datos, la transformación de datos, la adaptación de QoS y la mejora de escalabilidad del *discovery* DDS en escenarios de gran escala. Esta Tesis ha abordado estos problemas.

En el Capítulo 3 se propuso DDS-IS, una solución para la interconexión de espacios de datos DDS. Además, DDS-IS afronta tres cuestiones actualmente no resueltas en DDS: la transformación de datos entre dominios, la escalabilidad, y la adaptación de QoS entre entidades remotas. Además, y siguiendo la filosofía centrada en datos de DDS, el servicio propuesto permite realizar filtrado basado en

contenido de los datos intercambiados.

El diseño de DDS-IS permite una transformación de datos eficiente, que se aplica transparentemente durante la transmisión de información entre aplicaciones. En consecuencia, múltiples versiones de una misma aplicación, o incluso aplicaciones de diferentes proveedores, pueden interoperar de forma directa y sin requerir cambios, pese a tener diferentes estructuras de datos e interfaces. DDS-IS también soporta el filtrado basado en contenido, que puede ser usado para evitar que información sensible de un espacio de datos se filtre a otros dominios.

Uno de los puntos fuertes de DDS-IS es que es totalmente compatible con las últimas especificaciones de DDS. En consecuencia, el diseño propuesto no es dependiente de implementación, y por tanto es aplicable a cualquier implementación de DDS actual o futura. En concreto, DDS-IS utiliza DDS-XTypes [71], una nueva especificación del OMG para acceder a tipos de tópicos descubiertos en tiempo de ejecución.

Un prototipo de DDS-IS ha sido implementado para demostrar las ventajas de nuestra propuesta. Utilizando este prototipo, hemos presentado unos resultados obtenidos experimentalmente para evaluar el impacto del servicio propuesto en las prestaciones de DDS. Los resultados obtenidos demuestran que DDS-IS tiene un impacto muy reducido en términos de retardo y *throughput*. También se ha demostrado que DDS-IS puede ayudar a mejorar la escalabilidad de despliegues basados en DDS, ya que reduce el tráfico de red entre los espacios de datos interconectados. Por último, se ha conducido un estudio del impacto del servicio en la provisión en las políticas de QoS de DDS. De este estudio se concluye que la provisión de QoS es únicamente aplicable entre entidades en un mismo dominio (*i.e.*, salto a salto) para la mayor parte de políticas de QoS. Sin embargo, se han identificado dos excepciones: LIVELINESS y LIFESPAN. Estas dos políticas de QoS pueden ser provistas a través de varios dominios interconectados (*i.e.*, extremo a extremo).

DDS-IS no aborda el problema de la identificación universal de contenidos y entidades, ni el establecimiento de sesiones extremo a extremo, ni la escalabilidad del descubrimiento DDS en escenarios de gran escala. Con objeto de resolver estos problemas, en el Capítulo 4 se define un mecanismo para la identificación universal de contenidos y entidades DCPS, y se propone el diseño de un protocolo para la señalización de sesiones en entornos DCPS. Este protocolo permite el descubrimiento de entidades y contenidos en entornos de media escala. Además, permite la creación de sesiones entre nodos, lo que facilita la creación de rutas DCPS y la provisión de QoS extremo a extremo.

En este momento, el protocolo aún no ha sido implementado. Sin embargo, uno de los principales implementadores de DDS, RTI Inc. [89], ha mostrado su interés por implementar DCSIP, lo que augura un futuro muy prometedor para la propuesta.

Pese a sus ventajas, esta solución no es adecuada para escenarios DCPS de gran escala, dado que la presencia de un servidor de *rendezvous* no escala bien en despliegues con varios miles de nodos. Para solucionar este problema, en el Capítulo 5 se propone una solución escalable basada en P2P para el *discovery* y la transferencia de datos.

Nuestra solución se basa en el *framework* RELOAD, uno de los últimos estándares de la IETF. En este sentido, esta Tesis propone una arquitectura basada en RELOAD para desplegar escenarios IoT basados en DCPS. Además, se definen tres nuevos *Kinds* que extienden la funcionalidad de RELOAD, y que se corresponden a tres tipos de entidades DCPS: participantes, publicadores y suscriptores.

El sistema propuesto resuelve el *rendezvous* para sensores, actuadores y aplicaciones que comparten un espacio de datos común o un tópico de interés. Una vez se completa el *rendezvous*, los nodos utilizan la funcionalidad de RELOAD para establecer una conexión. Las aplicaciones de usuario pueden entonces usar dicha conexión para transmitir datos mediante una aproximación publicación-suscripción. A este respecto, nuestra solución tiene la ventaja adicional de permitir el paso a través de NATs con ICE.

Nuestra propuesta es extensible y cualquier otra aplicación publicación-suscripción puede utilizarla, como las basadas en CoAP o en DDS.

Con objeto de validar nuestra propuesta y evaluar su escalabilidad y robustez, se han conducido una serie de experimentos sobre redes simuladas de 500 a 1000 nodos. La topología de estas redes está basada en datos reales. En concreto, el entorno experimental se configuró utilizando datos de PlanetLab.

Los resultados experimentales obtenidos demuestran que nuestra propuesta es escalable. En particular, se ha observado un crecimiento logarítmico (función del tamaño de la red) del retardo medio para las principales operaciones del sistema. Con respecto a su robustez, se ha estresado una red de 5000 nodos con diferentes niveles de *churn*. Los resultados obtenidos muestran que, incluso para un elevado nivel de *churn* (se ha estresado el sistema con niveles equivalentes a cambiar un 25% de los nodos de la red cada 10 minutos) el porcentaje de errores en la localización de recursos es inferior al 18.17%.

Para finalizar esta sección, a continuación se revisan los objetivos enumerados en el Capítulo 1, y se examina en qué grado han sido cumplidos dichos objetivos.

- **Interconexión de espacios de datos DDS:**

- **Objetivo inicial:** Diseñar un servicio de interconexión de espacios de datos DDS.

- * **Resultados:** En colaboración con RTI, se ha diseñado DDS-IS y solicitado una patente con número US 2011/0295923 A1 [18]. Además, el diseño preliminar fue presentado en el Real-Time OMG Workshop 2009 [76].
 - **Objetivo inicial:** Incrementar la extensibilidad y flexibilidad de los sistemas DDS, facilitando la evolución de los sistemas basados en dicho estándar.
 - * **Resultados:** Aprovechando la funcionalidad de la especificación DDS-XTypes, DDS-IS incorpora la capacidad de efectuar transformaciones de los datos basadas en contenido. DDS-IS también soporta la integración de dominios DDS con diferentes políticas de QoS.
 - **Objetivo inicial:** Implementar y evaluar el servicio diseñado.
 - * **Resultados:** Se ha implementado un prototipo de DDS-IS y se han conducido una serie de experimentos para medir las prestaciones y escalabilidad del servicio. Los resultados de esta evaluación fueron publicados en una revista indexada en el JCR [53].
 - **Objetivo inicial:** Estudiar el impacto del servicio propuesto sobre las políticas de QoS de DDS.
 - * **Resultados:** Se ha conducido un análisis del impacto de DDS-IS en las políticas de QoS de DDS. En concreto, se han identificado cuáles de esas políticas pueden ser provistas salto a salto y cuáles extremo a extremo, de acuerdo a la especificación de DDS existente.
- **Señalización de sesión en DDS:**
 - **Objetivo inicial:** Diseñar un formato de identificación universal de entidades y contenidos en entornos DCPS.
 - * **Resultados:** Se ha propuesto el concepto de FQDN para la identificación de dominios, FQTN para la identificación de tópicos, DCPS-Entity-Group para la identificación de grupos de entidades del mismo tipo (participante, publicador, suscriptor) asociadas a un FQDN o FQTN concreto, y DCPS-Entity-Name para la identificación de entidades DCPS. Esta propuesta se encuentra actualmente en proceso de revisión en una revista indexada en el JCR.
 - **Objetivo inicial:** Diseñar un protocolo para el descubrimiento de entidades DCPS y para el establecimiento de sesión en DDS.
 - * **Resultados:** En el OMG Real-Time Workshop 2010 se presentó un trabajo en el que se proponía utilizar SIP para el establecimiento de sesiones DDS [51]. Este trabajo derivó posteriormente en dos soluciones diferentes: en primer lugar, un protocolo de señalización de

sesiones centradas en datos denominado DCSIP; en segundo lugar, una extensión (*usage*) DCPS para RELOAD. En esta Tesis se ha presentado el diseño DCSIP y el diseño e implementación de un prototipo de la extensión propuesta. El diseño de dicha extensión se encuentra actualmente en proceso de revisión en una revista indexada en el JCR.

- **Objetivo inicial:** Evaluar la escalabilidad y robustez de la solución propuesta.
- * **Resultados:** Se han realizado una serie de experimentos que demuestran la escalabilidad y robustez de la extensión propuesta. Estos resultados se encuentran actualmente en proceso de revisión en una revista indexada en el JCR.

Además de los objetivos iniciales, se ha cumplido el siguiente objetivo adicional:

- **Usage de CoAP para RELOAD:**

- **Objetivo:** Proponer una extensión de RELOAD que soporte la conexión a nodos de reducidas prestaciones utilizando un modelo solicitud-respuesta.
- * **Resultados:** En colaboración con ERICSSON, se ha enviado un *draft* a la IETF denominado *CoAP usage for RELOAD* [45]. Este *draft* define una extensión a RELOAD que proporciona un servicio de *rendezvous* para nodos CoAP y soporta el *caching* de la información de los sensores. Este *draft* se presentó en la IETF 83 Meeting [1] y se encuentra actualmente en proceso de discusión en el P2PSIP WG [36].

7.2 Trabajo futuro

Como trabajo futuro, queremos desarrollar nuevas extensiones y explorar nuevas aplicaciones para DDS-IS. Por ejemplo, estamos interesados en estudiar cómo el protocolo de descubrimiento basado en filtros de Bloom propuesto en [92] afecta a la escalabilidad de escenarios basados en DDS-IS.

Las arquitecturas de *Cloud computing* han ganado recientemente el interés de la industria y del mundo académico. En este sentido, han surgido proyectos que se apoyan en DDS para el acceso a información de monitorización de sistemas de *Cloud*. Sin embargo, estas soluciones son muy dependientes de la implementación utilizada. Como trabajo futuro, nos gustaría utilizar el DDS-IS para integrar aplicaciones basadas en DDS en escenarios de *Cloud*. Dado que DDS-IS permite a las

aplicaciones usar tipos no conocidos en tiempo de compilación, creemos que su uso puede facilitar la creación de infraestructuras genéricas y flexibles de *Cloud*.

En esta Tesis se ha propuesto DCSIP, un protocolo de señalización de sesión en entornos DDS. Tenemos previsto implementar la API de DCSIP en colaboración con RTI. Después de su implementación, queremos integrar dicha API en el DDS-IS. Una vez finalice la integración de DDS-IS y DCSIP, estamos interesados en integrar las dos soluciones de descubrimiento propuestas en esta Tesis: la solución basada en DCSIP, y la solución basada en RELOAD. Una vez ambas arquitecturas sean integradas, será posible crear federaciones de dominios DDS autoconfigurables que se adapten a escenarios DDS de media y gran escala.

Con respecto al *usage* DCPS para RELOAD propuesto, queremos centrar nuestra investigación en introducir mecanismos de distribución de carga en escenarios con un gran número de entidades por DCPS-Entity-Group (*i.e.*, escenarios donde el número de entradas asociadas a un único *Resource-ID* es alto). Por último, tenemos previsto proponer para su estandarización el *usage* DCPS para RELOAD propuesto en esta Tesis.

Bibliography

- [1] *IETF 83 - Paris, France*, 2012. URL <http://www.ietf.org/meeting/83/index.html>.
- [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] AMQP. Advanced Message Queuing Protocol. 2013. URL <http://www.amqp.org/>.
- [4] Apache Foundation. Apache QPid, 2011. URL <http://qp.id.apache.org/>.
- [5] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, October 2010. ISSN 13891286. doi: 10.1016/j.comnet.2010.05.010. URL <http://dx.doi.org/10.1016/j.comnet.2010.05.010>.
- [6] Francois Audet. The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP). Technical Report 5630, RFC Editor, Fremont, CA, USA, October 2009. URL <http://www.rfc-editor.org/rfc/rfc5630.txt>.
- [7] Guruduth Banavar, Tushar Chandra, Robert Strom, and Daniel Sturman. A case for message oriented middleware. In *In Proceedings of the 13th International Symposium on Distributed Computing*, pages 1–18, 1999. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.9592>.
- [8] David A. Bryan, Bruce B. Lowekamp, and Cullen Jennings. SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System. In *Advanced Architectures and Algorithms for Internet Delivery and Applications, 2005. AAA-IDEA 2005. First International Workshop on*, pages 42–49. IEEE, June 2005.

- ISBN 0-7695-2525-3. doi: 10.1109/aaa-idea.2005.15. URL <http://dx.doi.org/10.1109/aaa-idea.2005.15>.
- [9] David A. Bryan, Philip Matthews, Eunsoo Shim, Spencer Dawkins, and Dean Willis. Concepts and Terminology for Peer to Peer SIP. Technical Report draft-ietf-p2psip-concepts-04.txt, IETF Secretariat, Fremont, CA, USA, October 2011. URL <http://www.rfc-editor.org/internet-drafts/draft-ietf-p2psip-concepts-04.txt>.
- [10] Rajkumar Buyya, Chee S. 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://dx.doi.org/10.1016/j.future.2008.12.001>.
- [11] Fengyun Cao and Jaswinder P. Singh. Efficient event routing in content-based publish-subscribe service networks. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 929–940 vol.2. IEEE, March 2004. ISBN 0-7803-8355-9. doi: 10.1109/INFCOM.2004.1356980. URL <http://dx.doi.org/10.1109/INFCOM.2004.1356980>.
- [12] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, August 2001. ISSN 0734-2071. doi: 10.1145/380749.380767. URL <http://dx.doi.org/10.1145/380749.380767>.
- [13] Antonio Carzaniga, Matthew J. Rutherford, and Alexander L. Wolf. A Routing Scheme for Content-Based Networking. In *Proceedings of IEEE INFOCOM 2004*, Hong Kong, China, March 2004.
- [14] G. Chen, M. Li, and D. Kotz. Data-centric middleware for context-aware pervasive computing. *Pervasive and Mobile Computing*, 4(2):216–253, April 2008. doi: 10.1016/j.pmcj.2007.10.001. URL <http://linkinghub.elsevier.com/retrieve/pii/S1574119207000636>.
- [15] Yongjin Choi, Keuntae Park, and Daeyeon Park. A peer-to-peer overlay architecture for large-scale content-based publish/subscribe systems. In *in Third International Workshop on Distributed Event-Based Systems*, 2004. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.202.8833>.
- [16] A. Corradi and L. Foschini. A DDS-compliant P2P infrastructure for reliable and QoS-enabled data dissemination. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, volume 0, pages

- 1–8, Los Alamitos, CA, USA, May 2009. IEEE. ISBN 978-1-4244-3751-1. doi: 10.1109/ipdps.2009.5160957. URL <http://dx.doi.org/10.1109/ipdps.2009.5160957>.
- [17] Antonio Corradi, Luca Foschini, and Luca Nardelli. A DDS-compliant infrastructure for fault-tolerant and scalable data dissemination. In *IEEE Symposium on Computers and Communications (ISCC), 2010*, pages 489–495, 2010. doi: 10.1109/ISCC.2010.5546756. URL <http://dx.doi.org/10.1109/ISCC.2010.5546756>.
- [18] Alejandro De-Campos-Ruiz, Gerardo Pardo-Castellote, Jose M. Lopez-Vega, and Fernando Crespo-Sanchez. Patent US20110295923 - Bridging data distribution services domains based on discovery data. Technical report, May 2011. URL <http://www.google.com/patents/US20110295923>.
- [19] Edwin de Jong. RTI Eases Scaling and Integration of Real-Time Systems Across WANs and Systems of Systems. In *Embedded Systems Conference*, September 2009. URL <http://www.rti.com/company/news/dds-routing-service.html>.
- [20] Vladimir Dimitrov and Ventzislav Koptchev. PSIRP project – publish-subscribe internet routing paradigm: new ideas for future internet. In *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies, CompSysTech '10*, pages 167–171, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0243-2. doi: 10.1145/1839379.1839409. URL <http://dx.doi.org/10.1145/1839379.1839409>.
- [21] Patrick T. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne M. Ker-marrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2): 114–131, June 2003. ISSN 0360-0300. doi: 10.1145/857076.857078. URL <http://dx.doi.org/10.1145/857076.857078>.
- [22] Nikos Fotiou, Dirk Trossen, and GeorgeC Polyzos. Illustrating a publish-subscribe Internet architecture. *Telecommunication Systems*, 51(4):233–245, February 2012. ISSN 1018-4864. doi: 10.1007/s11235-011-9432-5. URL <http://dx.doi.org/10.1007/s11235-011-9432-5>.
- [23] Geoffrey Fox, Sang Lim, Shrideep Pallickara, and Marlon Pierce. Message-Based Cellular Peer-to-Peer Grids: Foundations for Secure Federation and Autonomic Services. *Future Generation Computer Systems*, 21(3):401–415, 2005. doi: 10.1016/j.future.2004.04.010. URL <http://dx.doi.org/10.1016/j.future.2004.04.010>.

- [24] John F. Gantz. The Expanding Digital Universe: a forecast of worldwide information growth through 2010. Technical report, International Data Corporation, March 2007. URL <http://www.emc.com/collateral/analyst-reports/expanding-digital-idc-white-paper.pdf>.
- [25] Abhishek Gupta, Ozgur D. Sahin, Divyakant Agrawal, and Amr El Abbadi. Meghdoot: Content-Based Publish/Subscribe over P2P Networks. In Hans-Arno Jacobsen, editor, *Middleware 2004*, volume 3231 of *Lecture Notes in Computer Science*, pages 254–273. Springer Berlin Heidelberg, 2004. doi: 10.1007/978-3-540-30229-2_14. URL http://dx.doi.org/10.1007/978-3-540-30229-2_14.
- [26] Akram Hakiri, Aniruddha Gokhale, Douglas C. Schmidt, Berthou Pascal, Joe Hoffert, and Gayraud Thierry. A SIP-Based Network QoS Provisioning Framework for Cloud-Hosted DDS Applications. In Robert Meersman, Tharam Dillon, Pilar Herrero, Akhil Kumar, Manfred Reichert, Li Qing, Beng-Chin Ooi, Ernesto Damiani, Douglas C. Schmidt, Jules White, Manfred Hauswirth, Pascal Hitzler, and Mukesh Mohania, editors, *On the Move to Meaningful Internet Systems: OTM 2011*, volume 7045 of *Lecture Notes in Computer Science*, chapter 7, pages 507–524. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-25105-4. doi: 10.1007/978-3-642-25106-1_7. URL http://dx.doi.org/10.1007/978-3-642-25106-1_7.
- [27] ZhenWu Hao, Wei Wang, Yu Meng, and YongLin Peng. An SNMP Usage for RELOAD. Technical Report draft-peng-p2psip-snmpp-02.txt, IETF Secretariat, Fremont, CA, USA, July 2011. URL <http://www.rfc-editor.org/internet-drafts/draft-peng-p2psip-snmpp-02.txt>.
- [28] Klaus Hartke. Observing Resources in CoAP. Technical Report draft-ietf-core-observe-05.txt, IETF Secretariat, Fremont, CA, USA, March 2012. URL <http://www.rfc-editor.org/internet-drafts/draft-ietf-core-observe-05.txt>.
- [29] Klaus Hartke. Observing Resources in CoAP. Technical Report draft-ietf-core-observe-08.txt, IETF Secretariat, Fremont, CA, USA, February 2013. URL <http://www.rfc-editor.org/internet-drafts/draft-ietf-core-observe-08.txt>.
- [30] icoup. MicroDDS, Data Distribution Service (DDS) for embedded systems, 2013. URL <http://www.icoup-consulting.com/microdds.html>.
- [31] IEC. "IEC PAS 62030 - Real-Time Publish-Subscribe (RTPS) Wire Protocol Specification Version 1.0". Technical report, IEC, November 2004. URL http://webstore.iec.ch/p-preview/info_iecpas62030%7Bed1.0%7Den.pdf.
- [32] IEC. International Electrotechnical Commission, 2013.

- [33] IETF. Multiparty Multimedia Session Control (mmusic) - Charter, 1993. URL <http://datatracker.ietf.org/wg/mmusic/charter/>.
- [34] IETF. SIP for Instant Messaging and Presence Leveraging Extensions (simple) - Charter, 2001. URL <http://datatracker.ietf.org/wg/simple/charter/>.
- [35] IETF. Session Initiation Proposal Investigation (sipping) - Charter, 2001. URL <http://datatracker.ietf.org/wg/sipping/charter/>.
- [36] IETF. Peer-to-Peer Session Initiation Protocol (p2psip), 2007. URL <http://datatracker.ietf.org/wg/p2psip/charter/>.
- [37] IETF. Session Initiation Protocol Core (sipcore) - Charter, 2009. URL <http://datatracker.ietf.org/wg/sipcore/charter/>.
- [38] IETF. Constrained RESTful Environments (core) - Charter, 2010. URL <http://datatracker.ietf.org/wg/core/charter/>.
- [39] IETF. Internet Engineering Task Force (IETF), 2013.
- [40] V. Jacobson and M. Handley. SDP: Session Description Protocol. Technical Report 2327, RFC Editor, Fremont, CA, USA, April 1998. URL <http://www.rfc-editor.org/rfc/rfc2327.txt>.
- [41] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies, CoNEXT '09*, pages 1–12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-636-6. doi: 10.1145/1658939.1658941. URL <http://dx.doi.org/10.1145/1658939.1658941>.
- [42] Cullen Jennings, Jonathan Rosenberg, and Marc Petit-Huguenin. A Usage of Resource Location and Discovery (RELOAD) for Public Switched Telephone Network (PSTN) Verification. Technical Report draft-petithuguenin-vipr-reload-usage-02.txt, IETF Secretariat, Fremont, CA, USA, July 2011. URL <http://www.rfc-editor.org/internet-drafts/draft-petithuguenin-vipr-reload-usage-02.txt>.
- [43] Cullen Jennings, Salman Baset, Henning Schulzrinne, Bruce Lowekamp, and Eric Rescorla. A SIP Usage for RELOAD. Technical Report draft-ietf-p2psip-sip-07.txt, IETF Secretariat, Fremont, CA, USA, January 2012. URL <http://www.rfc-editor.org/internet-drafts/draft-ietf-p2psip-sip-07.txt>.
- [44] Cullen Jennings, Bruce B. Lowekamp, Eric Rescorla, Salman A. Baset, and Henning Schulzrinne. REsource LOcation And Discovery (RELOAD)

- Base Protocol. Technical Report draft-ietf-p2psip-base-24.txt, IETF Secretariat, Fremont, CA, USA, January 2013. URL <http://www.rfc-editor.org/internet-drafts/draft-ietf-p2psip-base-24.txt>.
- [45] Jaime Jimenez, Jose M. Lopez-Vega, Jouni Maenpaa, and Gonzalo Camarillo. A Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD). Technical Report draft-jimenez-p2psip-coap-reload-03.txt, IETF Secretariat, Fremont, CA, USA, February 2013. URL <http://www.rfc-editor.org/internet-drafts/draft-jimenez-p2psip-coap-reload-03.txt>.
- [46] Paul E. Jones and 3rd. US Secure Hash Algorithm 1 (SHA1). Technical Report 3174, RFC Editor, Fremont, CA, USA, September 2001. URL <http://www.rfc-editor.org/rfc/rfc3174.txt>.
- [47] Kongsberg Gruppen. InterCOM DDS, 2013. URL <http://www.kongsberg.com/en/kds/kongsberggallium/products/intercom%20dds/>.
- [48] Wolfgang Lehner and Wolfgang Hummer. The revolution ahead: Publish/subscribe meets database systems. July 2009. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.89.7093>.
- [49] 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. In *Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems*, 2012.
- [50] Jose M. Lopez-Vega, Javier Sanchez-Monedero, Javier Povedano-Molina, and Juan M. Lopez-Soler. QoS Policies for Audio/Video Distribution Over DDS Middleware. In *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, July 2008. URL http://www.omg.org/news/meetings/workshops/rt_embedded_2008.htm.
- [51] Jose M. Lopez-Vega, Javier Povedano-Molina, and Juan M. Lopez-Soler. DDS/SIP Interworking: A DDS-SIP Gateway. In *Real-time and Embedded Systems Workshop*, May 2010. URL <http://www.omg.org/news/meetings/SMCS/rt/Program.htm>.
- [52] Jose M. Lopez-Vega, Javier Povedano-Molina, Javier Sanchez-Monedero, and Juan M. Lopez-Soler. Políticas de QoS en una Plataforma de Trabajo Colaborativo sobre Middleware DDS. In *XIII Jornadas de Tiempo Real*, February 2010.
- [53] Jose M. Lopez-Vega, Javier Povedano-Molina, Gerardo Pardo-Castellote, and Juan M. Lopez-Soler. A content-aware bridging service for publish/subscribe environments. *Journal of Systems and Software*, 86(1):108–124, January 2013.

ISSN 01641212. doi: 10.1016/j.jss.2012.07.033. URL <http://dx.doi.org/10.1016/j.jss.2012.07.033>.

- [54] J. Maenpaa and G. Camarillo. Study on maintenance operations in a chord-based Peer-to-Peer session initiation protocol overlay network. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–9. IEEE, May 2009. ISBN 978-1-4244-3751-1. doi: 10.1109/ipdps.2009.5160949. URL <http://dx.doi.org/10.1109/ipdps.2009.5160949>.
- [55] J. Maenpaa and G. Camarillo. Analysis of Delays in a Peer-to-Peer Session Initiation Protocol Overlay Network. In *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pages 1–6. IEEE, January 2010. ISBN 978-1-4244-5175-3. doi: 10.1109/ccnc.2010.5421852. URL <http://dx.doi.org/10.1109/ccnc.2010.5421852>.
- [56] J. Maenpaa, V. Andersson, G. Camarillo, and A. Keranen. Impact of Network Address Translator Traversal on Delays in Peer-to-Peer Session Initiation Protocol. In *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, pages 1–6. IEEE, December 2010. ISBN 978-1-4244-5636-9. doi: 10.1109/glocom.2010.5683295. URL <http://dx.doi.org/10.1109/glocom.2010.5683295>.
- [57] Jouni Maenpaa. Performance evaluation of Recursive Distributed Rendezvous based service discovery for Peer-to-Peer Session Initiation Protocol. *Computer Networks*, 56(5):1612–1626, March 2012. ISSN 13891286. doi: 10.1016/j.comnet.2012.01.015. URL <http://dx.doi.org/10.1016/j.comnet.2012.01.015>.
- [58] Jouni Maenpaa and J. J. Bolonio. Performance of REsource LOcation and Discovery (RELOAD) on Mobile Phones. In *Wireless Communications and Networking Conference (WCNC), 2010*, pages 1–6. IEEE, April 2010. ISBN 978-1-4244-6396-1. doi: 10.1109/wcnc.2010.5506653. URL <http://dx.doi.org/10.1109/wcnc.2010.5506653>.
- [59] Jouni Maenpaa and Gonzalo Camarillo. Estimating operating conditions in a Peer-to-Peer Session Initiation Protocol overlay network. In *24th IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2010)*, pages 1–8, April 2010. doi: 10.1109/ipdpsw.2010.5470935. URL <http://dx.doi.org/10.1109/ipdpsw.2010.5470935>.
- [60] Gregory Marsh, A. P. Sampat, Sreeram Potluri, and D. K. Panda. Scaling Advanced Message Queuing Protocol (AMQP) Architecture with Broker Federation and InfiniBand. Technical report, 2010. URL <ftp://ftp.cse.ohio-state.edu/pub/tech-report/2009/TR17.pdf>.

- [61] MilSOFT. MilSOFT DDS Middleware, 2013. URL <http://dds.milsoft.com.tr/en/dds-home.php>.
- [62] Object Computing, Inc. OpenDDS — Object Computing, Inc., 2013. URL <http://www.ocieweb.com/products/opensdds>.
- [63] OCERA. OCERA ORTE, an open source implementation of RTPS, 2013. URL <http://www.ocera.org/download/components/WP7/orte-0.3.1.html>.
- [64] The Trustees of Princeton University. PlanetLab - An open platform for developing, deploying, and accessing planetary-scale services, 2013. URL <http://www.planet-lab.org/>.
- [65] Sangyoon Oh, Jai-Hoon Kim, and Geoffrey Fox. Real-time performance analysis for publish/subscribe systems. *Future Generation Computer Systems*, 26(3): 318–323, March 2010. ISSN 0167739X. doi: 10.1016/j.future.2009.09.001. URL <http://dx.doi.org/10.1016/j.future.2009.09.001>.
- [66] T. Okabe and H. Schulzrinne. Peer-to-Peer SIP Features to Eliminate a SIP Sign-Up Process. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5, 2008. doi: 10.1109/glocom.2008.ecp.340. URL <http://dx.doi.org/10.1109/glocom.2008.ecp.340>.
- [67] OMG. Data-Distribution Service for Real-Time Systems (DDS).v1.2. Technical report, OMG, 2006. URL <http://www.omg.org/cgi-bin/doc?formal/07-01-01.pdf>.
- [68] OMG. Common Object Request Broker Architecture (CORBA/IIOP).v3.1. Technical report, OMG, January 2008. URL <http://www.omg.org/spec/CORBA/3.1/>.
- [69] OMG. The Real-Time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification, V2.1. Technical report, OMG, November 2010. URL <http://www.omg.org/spec/DDS-RTPS/2.1/>.
- [70] OMG. Extensible And Dynamic Topic Types For DDS (DDS-XTypes) 1.0 (FTF 2) - Beta 2. Technical report, 2011. URL <http://www.omg.org/spec/DDS-XTypes/1.0/Beta2/>.
- [71] OMG. Extensible And Dynamic Topic Types For DDS (DDS-XTypes). Technical report, November 2012. URL <http://www.omg.org/spec/DDS-XTypes/>.
- [72] OMG. List of DDS vendors, 2013. URL <http://portals.omg.org/dds/category/web-links/vendors>.
- [73] OMG. Object Management Group, 2013. URL <http://www.omg.org/>.

- [74] ORACLE. Java Message Service (JMS), October 2010. URL <http://www.oracle.com/technetwork/java/index-jsp-142945.html>.
- [75] G. Pardo-Castellote. OMG Data-Distribution Service: Architectural Overview. In *23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)*, volume 0, pages 200–206, Los Alamitos, CA, USA, May 2003. IEEE Computer Society. ISBN 0-7695-1921-0. doi: 10.1109/icdcs.2003.1203555. URL http://portals.omg.org/dds/WhitepapersPage?action=AttachFile&do=get&target=DDS_Architectural_Overview.pdf.
- [76] Gerardo Pardo-Castellote, Fernando Sanchez, and Jose M. Lopez-Vega. Deploying DDS on a WAN and the GIG: The DDS Router. In *Real-time and Embedded Systems Workshop*. OMG, July 2009. URL <http://www.omg.org/news/meetings/GOV-WS/pr/rte.htm>.
- [77] Yunjung Park, Duckwon Chung, Dugki Min, and Eunmi Choi. Middleware Integration of DDS and ESB for Interconnection between Real-Time Embedded and Enterprise Systems. *Convergence and Hybrid Information Technology*, pages 337–344, 2011. URL <http://www.springerlink.com/index/U5POU30775807686.pdf>.
- [78] Javier Povedano-Molina, Jose M. Lopez-Vega, and Juan M. Lopez-Soler. EMDS: an Extensible Multimedia Distribution Service. In *Real-time and Embedded Systems Workshop*, May 2010. URL <http://www.omg.org/news/meetings/SMCS/rt/Program.htm>.
- [79] 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.
- [80] PRISMTECH. OpenSplice DDS Industry Solutions, 2009. URL <http://www.opensplice.com/section-item.asp?snum=2&sid=70>.
- [81] PRISMTECH. OpenSplice DDS Data Distribution Service for Real-Time Systems, 2013.
- [82] PURSUIT. Publish Subscribe Internet Technology (PURSUIT) project site. Technical report, 2010. URL http://fp7pursuit.ipower.com/PursuitWeb/wp-content/uploads/2011/03/PURSUIT_fact_sheet.pdf.
- [83] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC3261 - SIP: Session Initiation Protocol. Technical report, IETF, June 2002. URL <http://www.ietf.org/rfc/rfc3261.txt>.

- [84] Jonathan Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. Technical Report 5245, RFC Editor, Fremont, CA, USA, April 2010. URL <http://www.rfc-editor.org/rfc/rfc5245.txt>.
- [85] RTI. Meeting Real-Time Requirements in Integrated Defense Systems, 2007. URL http://www.rti.com/whitepapers/Real-Time_Integrated_Defense_Systems.pdf.
- [86] RTI. RTI Eases Integration of Large-Scale Real-Time Applications with High-Capacity Networked Sensors. In *Embedded Systems Conference*, August 2008. URL <http://www.rti.com/company/news/networked-sensors.html>.
- [87] RTI. RTI Industry Solutions, 2009. URL <http://www.rti.com/solutions/>.
- [88] RTI. Real-Time Innovations (RTI) DDS Data Distribution Service, 2012. URL <http://www.rti.com/>.
- [89] RTI. Connex DDS, 2013. URL <http://www.rti.com/>.
- [90] O. Sahin, C. Gerede, Divyakant Agrawal, A. El Abbadi, Oscar Ibarra, and Jianwen Su. Spider: P2P-based web service discovery. *Service-Oriented Computing-ICSOC 2005*, pages 157–169, 2005. URL <http://www.springerlink.com/index/p544hw13624630q6.pdf>.
- [91] Javier Sanchez-Monedero, Javier Povedano-Molina, Jose M. Lopez-Vega, and Juan M. Lopez-Soler. An XML-Based Approach to the Configuration and Deployment of DDS Applications. In *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, July 2008. URL http://www.omg.org/news/meetings/workshops/Real-time_WS_Final_Presentations_2008/Session%202/02-03_Monedero_et_al.pdf.
- [92] Javier Sanchez-Monedero, Javier Povedano-Molina, Jose M. Lopez-Vega, and Juan M. Lopez-Soler. Bloom filter based discovery protocol for DDS middleware. *Journal of Parallel and Distributed Computing*, 71(10):1305–1317, May 2011. ISSN 07437315. doi: 10.1016/j.jpdc.2011.05.001. URL <http://dx.doi.org/10.1016/j.jpdc.2011.05.001>.
- [93] J. M. Schlesselman, G. Pardo-Castellote, and B. Farabaugh. OMG Data-Distribution Service (DDS): Architectural update. volume 2, pages 961–967, 2004. URL <http://www.scopus.com/inward/record.url?eid=2-s2.0-27744565488&partnerID=40&md5=eb8c4bc899c186f494703bc48b1d06c5>. cited By (since 1996) 3.
- [94] Cristina Schmidt and Manish Parashar. A Peer-to-Peer Approach to Web Service Discovery. *World Wide Web*, 7(2):211–229, June 2004. ISSN 1386-145X.

- doi: 10.1023/b:wwwj.0000017210.55153.3d. URL <http://dx.doi.org/10.1023/b:wwwj.0000017210.55153.3d>.
- [95] Douglas C. Schmidt and Carlos O’Ryan. Patterns and performance of distributed real-time and embedded publisher/subscriber architectures. *Journal of Systems and Software*, 66(3):213–223, June 2003. ISSN 01641212. doi: 10.1016/s0164-1212(02)00078-x. URL [http://dx.doi.org/10.1016/s0164-1212\(02\)00078-x](http://dx.doi.org/10.1016/s0164-1212(02)00078-x).
- [96] Douglas C. Schmidt, David L. Levine, and Sumedh Mungee. The Design of the tao real-time object request broker. In *Computer Communications*, pages 294–324, 1998. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.119.2161>.
- [97] Zach Shelby, Klaus Hartke, Carsten Bormann, and Brian Frank. Constrained Application Protocol (CoAP). Technical Report draft-ietf-core-coap-13.txt, IETF Secretariat, Fremont, CA, USA, December 2012. URL <http://www.rfc-editor.org/internet-drafts/draft-ietf-core-coap-13.txt>.
- [98] Dongcai Shi, Jianwei Yin, Zhaohui Wu, and Jinxiang Dong. A Peer-to-Peer Approach to Large-Scale Content-Based Publish-Subscribe. In *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology, WI-IATW ’06*, pages 172–175, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2749-3. doi: 10.1109/wi-iatw.2006.18. URL <http://dx.doi.org/10.1109/wi-iatw.2006.18>.
- [99] Kundan Singh and Henning Schulzrinne. Peer-to-peer internet telephony using SIP. In *NOSSDAV ’05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 63–68, New York, NY, USA, 2005. ACM. ISBN 1-58113-987-X. doi: 10.1145/1065983.1065999. URL <http://dx.doi.org/10.1145/1065983.1065999>.
- [100] Sistemi Software Integrati. BEE DDS — Real-time Middleware scalable and secure inspired by the swarm intelligence, 2013. URL <http://www.beedds.com/en/index.html>.
- [101] Charilaos Stais, Dimitris Diamantis, Christina Aretha, and George Xylomenos. VoPSI: Voice over a Publish-Subscribe Internetwork. In *Future Network & Mobile Summit (FutureNetw), 2011*, pages 1–8. IEEE, June 2011. ISBN 978-1-4577-0928-9. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6095195.
- [102] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies,*

- architectures, and protocols for computer communications*, volume 31 of SIGCOMM '01, pages 149–160, New York, NY, USA, October 2001. ACM. ISBN 1-58113-411-8. doi: 10.1145/383059.383071. URL <http://dx.doi.org/10.1145/383059.383071>.
- [103] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari R. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for Internet applications. *Networking, IEEE/ACM Transactions on*, 11(1):17–32, February 2003. ISSN 1063-6692. doi: 10.1109/tnet.2002.808407. URL <http://dx.doi.org/10.1109/tnet.2002.808407>.
- [104] Wesley W. Terpstra, Stefan Behnel, Ludger Fiege, Andreas Zeidler, and Alejandro P. Buchmann. A peer-to-peer approach to content-based publish/subscribe. In *Proceedings of the 2nd international workshop on Distributed event-based systems, DEBS '03*, pages 1–8, New York, NY, USA, 2003. ACM. ISBN 1-58113-843-1. doi: 10.1145/966618.966627. URL <http://dx.doi.org/10.1145/966618.966627>.
- [105] Twin Oaks Computing, Inc. CoreDX DDS, 2013. URL <http://www.twinoakscomputing.com/>.
- [106] W3C. Extensible Markup Language (XML) 1.0 (Fifth Edition). Technical report, W3C, November 2008. URL <http://www.w3.org/TR/REC-xml/>.
- [107] Sanghyun Yoo, Jin H. Son, and Myoung H. Kim. An efficient subscription routing algorithm for scalable XML-based publish/subscribe systems. *Journal of Systems and Software*, 79(12):1767–1781, December 2006. ISSN 01641212. doi: 10.1016/j.jss.2006.03.039. URL <http://dx.doi.org/10.1016/j.jss.2006.03.039>.
- [108] Sanghyun Yoo, Jin H. Son, and Myoung H. Kim. A scalable publish/subscribe system for large mobile ad hoc networks. *Journal of Systems and Software*, 82(7):1152–1162, July 2009. ISSN 01641212. doi: 10.1016/j.jss.2009.02.020. URL <http://dx.doi.org/10.1016/j.jss.2009.02.020>.
- [109] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, Jim Thornton, Diana K. Smetters, Beichuan Zhang, Gene Tsudik, Kc Claffy, Dmitri Krioukov, Dan Massey, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, Patrick Crowley, and Edmund Yeh. Named Data Networking (NDN) Project NDN-0001. Technical report, October 2010. URL <http://www.parc.com/content/attachments/named-data-networking.pdf>.