Instant Messaging Interface for Data Distribution Service

Javier Povedano-Molina\textsuperscript{1}, Jose M. Lopez-Vega\textsuperscript{1}, Javier Sanchez-Monedero\textsuperscript{2}, and Juan M. Lopez-Soler\textsuperscript{1}

\textsuperscript{1}\{jpovedano,jmlv,juanma\}@ugr.es
Dpto. Teoría de la Señal, Telemática y Comunicaciones
Universidad de Granada
\textsuperscript{2}\textsuperscript{102amoj@uco.es}
Dpto. Informática y Análisis Numérico
Universidad de Córdoba

\textbf{Abstract.} Data Distribution Service (DDS) middleware has gained relevance in the last years. In most cases, the success of growing technologies are related to their integration with existing technologies. DDS interoperability with other technologies, such as Web Services, is an open issue for the DDS research community. This work presents a proposal of integration between Data Distribution Service and Instant Messaging services. The idea is to let the user access to data being published in DDS with a regular instant messaging client using text messages. Also, the user can publish data in DDS using the same mechanism of instant text messages. This new interface between the user and DDS makes the communication more natural and intuitive. The presented architecture, also makes possible that DDS topics could be published across existing instant messaging networks, extending DDS functionality in Wide-Area Networks (WAN) without needing dedicated services deployments.

\textbf{Key words:} real-time, data-centric, user interfaces, Data Distribution Service, middleware, publish, subscribe

1 Introduction

Fully Distributed systems are architectures that are gained popularity in the recent years due to the evolution of devices and networks. Data Distribution Service (DDS)[\textsuperscript{1}]\textsuperscript{2} is a middleware oriented to build fully distributed systems with minimum effort. The data-centric publish-subscribe paradigm used by the DDS middleware achieves remarkable benefits when deploying complex architectures. The latest research efforts in DDS are aimed to improve the interoperability with other systems, such as Web Services, Corba and Java Messaging System (JMS). This work focuses on the DDS and Instant Messaging services interoperability issue.

The proposed approach is aimed to improve the communication between the final user and the DDS middleware. The user can access to the data being published in DDS with their instant messaging client in a natural way. The presented work can also be used as a monitor or debugging tool for DDS applications.
This document is structured as follows:

- In section 1 an introduction to the project and related basic concepts are explained.
- Data Distribution Service is reviewed in section 2.
- In section 3 a review of the objectives of this project is done.
- In Section 4 explains the design of the proposed architecture.
- Some details about implementation are detailed in section 5.
- Main conclusions are finally provided in section 6.

2 Data Distribution Service Middleware

Data Distribution Service (DDS) is a standard middleware proposed by the Object Management Group[3] for real-time publish/subscribe applications. DDS adopts a data-centric model, which is aimed to provide publish subscribe based services for high performance real-time tasks. One of the main advantages of this approach is that information producers and consumers are spatially and temporally decoupled.

According with the DDS standard, some key concepts are briefly defined for better understanding of this document:

Global Data Space An abstraction that represents all the data being published in DDS.

Domain Virtual network abstraction. Applications only share data if they are in the same domain. It is a isolation and communications optimization mechanism.

Domain Participant Abstraction that represents the participation of an application on a certain domain.

Topic Logic channel or minimum data exchange unit in DDS. A topic is defined by its name and data type.

Publisher Object responsible of disseminate data to appropriate subscribers.

Subscriber Object that actually takes the topics received by the middleware and delivers them to the application.

Data Writer Application entry point to the publisher for a specific kind of topic. Application publish topic samples by means of a Data Writer.

Data Reader Application entry point to a subscriber for a specific kind of topic. The Data Readers notify the application when there are new topic samples available.

Endpoints A generic way to denominate both Data Writers and Data Readers.

Quality of Service (QoS) Parameters that configure the DDS entities behavior. They are aimed to control local resources and certain communication aspects.

The DDS publish/subscribe paradigm is shown in figure 1. In this figure some relevant aspects about DDS can be remarked.

- There is a shared domain that can be viewed as a bus among all the DDS entities.
An application can publish or subscribe data which are being published in a given DDS domain.

Access to a domain is done by specialized entities (Endpoints). Each of them only knows how to handle a specific kind of topic.

Data producers (Publishers) publish samples of given topics by means of Data Writers, and consumers (Subscribers) do it by Data Readers.

Quality of Service (QoS) policies allow the user to control how the data distribution is achieved.

![Data Distribution Service](image)

**Fig. 1.** Data Distribution Service.

### 3 Objectives

In this work we propose an architecture that enables the user to access Data Distribution Service global data-space, with an interface based on instant messaging concepts. The main objectives to achieve are:

- Providing access to data being published in a DDS Global Data Space from an user friendly interface.
- Receiving notifications about DDS status changes synchronously using Instant messaging concepts such as presence, status and messages.
- Providing a bridge between DDS and Instant Messaging services.

To achieve these objectives the following features had been included in the proposed design:

**Automatic Discovery** The Entities that are present in the given DDS domain are discovered by the Instant Messenger client using the Builtin Topics facility provided by the middleware. This information is stored in XML trees for easing the processing and formatting of future user queries.
**JSON Support**  All the information provided by DDS is transformed into JSON format. Other output formats can also be easily added.

**Multiple Domains Support**  As the instant messenger application supports multiple accounts to be connected at the same time, we use this feature for supporting multiple DDS Domains.

**New Types Support**  The information about typecodes provided in the discovery process is used to support new topic types without re-compiling.

**Multiple Data Access Schemes**  The application can retrieve the information from *DataReader* asynchronously or after user polling requests.

## 4 DDS-IM Proposed Architecture

This section explains how the system is designed with special emphasis on component interactions. The architecture of the proposed system is depicted in figure 2. In this figure, the main modules that compound the system and their relationships/dependencies are shown.

![Fig. 2. Proposed Architecture](image)

DDS and Instant Messaging are two different paradigms that involves different concepts. The first step is to establish a mapping between this two *different worlds* in order to define the behavior of the application. In table 1 the proposed mapping has been summarized. According to this mapping, the system behavior can be summarized as:

1. The user “logs” in DDS using a certain account in a instant messaging client. Each account is associated to a DDS Domain.
Table 1. Proposed mapping between Data Distribution Service concepts and Instant Messaging concepts

<table>
<thead>
<tr>
<th>DDS</th>
<th>Instant Messaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainParticipant</td>
<td>Instant Messenger Account</td>
</tr>
<tr>
<td>Topic</td>
<td>Buddy</td>
</tr>
<tr>
<td>Domain</td>
<td>Buddy Group</td>
</tr>
</tbody>
</table>

2. For each remote discovered topic -it is done by the middleware-, a *buddy* appears in the user’s contact list interface. This buddy is the entry point for the user to interact with a given topic.

3. The user can access to the information relative to a topic by *chatting* with the associated buddy. Information can be related to the last received samples, the remote endpoints that handle this topic, and the status change of remote entities.

4. Synchronous notifications of middleware events are done by means of messages sent by buddies into a conversation window with the user.

For further details, the following sections briefly describe each of the proposed modules.

4.1 Instant Messaging Layer

This layer represents the mapping between network packets and their representation into a instant messaging client. According to the instant messaging protocol messages retrieved from the network, the contact list and conversations are updated. Also, this layer can be used to bridge the messages provided by our application to another instant messaging service (p.e. Jabber).

4.2 Discovery Module

This module is aimed to monitor all the entities present on a certain domain. The DDS builtin discovery process is accessed to get all the information available about topics that are being published in the Global Data Space. This information is summarized and mapped to instant messaging buddy status for each topic according to:

- **Offline** No publications about a certain topic had been discovered yet. This is the default status.
- **Online** At least one publication of a topic is alive.
- **Away** There are no active publications alive. The liveness of the last Data Writer for this topic had expired.
4.3 DDS XML Module

The information retrieved from the middleware is stored locally using native data types from the DDS API. These structures are difficult to process directly. In order to circumvallate this problem, all the middleware API structures are transformed to a XML format. With this approach, the main benefits are that XML format can be transformed easily into different representations using XSLT transformations\cite{4}, the document nodes can be accessed using the XPath\cite{5} standard and it is easy to do queries to a XML document using the XQuery\cite{6} standard. This module is also responsible of formatting XML representations into a more human-readable format.

4.4 DDS JSON Module

As many DDS structures, the topics samples are stored internally in a native format. In order to establish a relationship between the user and the middleware communication, a format to exchange this information has to be chosen. JavaScript Object Notation (JSON) is a lightweight and structured format that is easy to understand. This module takes JSON format strings and transforms them into topic samples (serialization) that can be delivered by the middleware and, vice-versa (deserialization) that can be delivered by the middleware.

4.5 Interpreter Module

The communication between the user and the DDS middleware is done using conversation windows, where the user writes about what he wants to know about the Global Data Space using a near-natural language grammar. For example, the user may ask about the remote publishers of a certain topic, the offered QoS for a certain publication or the last sample of a topic. This request is parsed by this module, and it is translated into DDS actions using the middleware API. The results are returned to the user in the same conversation window.

4.6 Dynamic Type Module

By default, the methods for handling a certain topic data type are hard coded into the application. Our application needs to access and handle topics that are not known a-priori. For this purpose a dynamic type module has to be created to handle any topic by means of their type code (information that is acquired during discovery).

4.7 Notification Module

When a event occurs in DDS middleware it might result in changes in the user interface. For example when a new topic publication has been discovered, the buddy that represents this topic in the contact list should change its state to online status. Another example is when a new topic sample arrives to a Data
Reader, the user should be notified with a new message in the conversation window associated to the topic that handles this Data Reader. This module is responsible of translating DDS events into changes in the user interface or instant messages.

5 Implementation

As a proof of concept, the presented work has been implemented using an existing open source instant messaging client called pidgin\cite{7}. A plugin has been developed to add DDS support to pidgin according to the proposed system architecture. In Figure 3, a couple of screenshots of our prototype are shown. The first image shows how an account is configured to be associated to a particular DDS domain. The second image shows a contact list with buddies that represent topics that had been discovered within a certain DDS Domain.

6 Conclusions

This work aims to improve the DDS interoperability. In particular we propose a new approach for DDS and Instant Messaging services interaction. The proposed architecture achieves that the DDS Global Data Space information can be accessed by a Instant Messenger application. Our system allows that chat clients can be used as a DDS monitoring and debugging tools.
One of the relevant aspects of this work is the fact that XML was chosen for internal representation formats. XML in conjunction with XSLT makes possible to translate the internal representation of DDS data to many instant messaging protocol formats (that usually are XML subsets). This approach allows the access to DDS Global Data Space by using existent Instant Messaging infrastructures and vice-versa. In this sense an important step for improving the DDS interoperability is provided.

References