Dedicated WS-DDS Interface for Sharing Information Between Civil and Military Domains

Przemysław Caban, Joanna Śliwa

Military Communication Institute, ul. Warszawska 22A, 05-130 Zegrze, Poland

Abstract: The article presents results of the study on the exchange of data between Web Services (WS) and Data Distribution Service (DDS) using WS-DDS Interface. WS-DDS Interface connects two architecturally different message exchange solutions dedicated for two different environments. Web Service is system-independent application very often used in over-provisioned network. DDS is designed for real time applications. It works in publish-subscribe mode providing efficient solution for resource constrained networks. The WS-DDS Interface presented in this article enables bi-directional traffic between WS and DDS, with regard to the timeframe of the protocol and data transformation, which is a very important factor in success of a mission.

Keywords: web service, SOAP message, SOA, tactical network, DDS

1. Introduction

Web service (WS) is one of the data transmission mechanisms dedicated for overprovisioned networks. WSs are independent of the operating system and implementation, flexible, reusable and enable composition of complex services from the basic ones, independently from their localization. What is more, changes in WS internal implementation do not impose changes in the interface specification. This enables hiding the service logic keeping exposed only the service interface. The listed advantages are, however, burdened with high web services transmissions overhead. This makes web services the most appropriate for overprovisioned communications systems usually used in civil organizations or on high command levels of military systems.

Data Distribution Service is dedicated for real – time systems. It enables the distribution of data from many sources to many destinations at the same time. Moreover, it does not require the network to be overprovisioned since the data overhead is rather small. It is very important in operational scenarios, where mobile users act dynamically and need to exchange information quickly. They usually use mobile wireless networks with limited resources, such as, e.g. tactical networks.

---

1 Work presented within the RTO IST-090 “SOA challenges over real time and disadvantaged grids”.
The specifics of these two technical (WS and DDS) domains hinder their interoperability. In order to exchange information it is necessary to provide a special interface that would enable connection of these domains, without limiting their independence and functionality.

WS-DDS Interface has been designed and implemented to solve this problem. It enables exchange of information between two technologically different domains – DDS domain and WS domain. This article presents the results of the study on the exchange of data between Web Services (WS) and Data Distribution Service (DDS) using WS-DDS Interface.

2. Web service and DDS data sharing comparison

Web service is a software component independent from the platform and the implementation, delivering definite functionality (.ies) through a unique, well described in a machine-processable format and easily accessible interface. The interface is described using Web Services Description Language (WSDL) [2]. Other systems interact with the Web service according to its description using the SOAP protocol.

Data Distribution Service for Real-time Systems (DDS) [3] is the first open international middleware standard directly addressing publish-subscribe communications for real-time and embedded systems created by Object Management Group (OMG) [4]. DDS introduces a virtual Global Data Space where applications can share information by simply reading and writing data-objects from so called Topics. DDS has an extensive control of the Quality of Service (QoS) parameters, including transport priority, reliability, history (possibility of data persistence within the topic in the DDS middleware), delivery deadlines, and resource limits.

Web service is built based on the client – server architecture (see Fig. 1).

![Client Server Architecture](image)

Figure 1. Client-Server Architecture

Application client (so called client stub) sends to the well-known address (so called endpoint) a request message and receives a response message. The major and largest disadvantage of this architecture is the lack of service accessibility when the application server is damaged, switched off or when some problems with the network connection have appeared.

DDS is built based on the Peer-to-Peer (P2P) architecture (see Fig. 2). DDS is asynchronous. It enables publishing of data in so called Topics. Data defined by
Topics is sent in one direction from Publishers to Subscribers. The main advantage of this architecture is loose coupling between DDS elements. If one of the participants publishing data is inaccessible, the DDS service is still accessible. Subscribers can receive data on the same topic from different publishers.

![P2P Architecture](image)

**Figure 2. P2P Architecture**

Table 1: WS and DDS comparison

<table>
<thead>
<tr>
<th>Compared object</th>
<th>Web Service</th>
<th>DDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural elements</td>
<td>– Provider, – Requester, – Broker (optional)</td>
<td>– Publisher, – Subscriber</td>
</tr>
<tr>
<td>Architecture</td>
<td>Client-Server</td>
<td>P2P</td>
</tr>
<tr>
<td>Message Exchange Patterns</td>
<td>Request-Response</td>
<td>Publish-Subscribe</td>
</tr>
<tr>
<td>Data flow</td>
<td>Remote method calls</td>
<td>One-way messages</td>
</tr>
<tr>
<td>Data model</td>
<td>SOAP Message</td>
<td>Data-Object</td>
</tr>
<tr>
<td>Data description</td>
<td>WSDL description</td>
<td>Topic description</td>
</tr>
<tr>
<td>QoS</td>
<td>--------</td>
<td>QoS Policy</td>
</tr>
</tbody>
</table>

The WS – DDS comparison has been shown in Table 1. Significant difference between DDS and Web Service is the possibility of creating, by the DDS application, Client subscriber object to the topic that have not started publishing yet. It is very important when the dynamics of an action in the battlefield is high. When one
of the battle participants begins publishing data, the DDS Global Space subscribers will receive it immediately. Web service technology does not enable creation of a client stub for receiving SOAP messages when the web service provider does not make the WSDL description accessible.

Web services are the most often used means of service oriented architecture (SOA) realization. WS related standards are being quickly applied in software development tools (also in the open source) which bring great benefit for the software developers. They are also being extended by additional recommendations and standards (thanks to OASIS and W3C organizations) which result in fast enhancement of their functionality. They are very often used in modern military command and control systems on high command levels to distribute information among different subsystems and users. They are also applied by the civil units that need easy and quickly implementable solutions.

However, WS utilization on low command levels, where mobile operators act in a dynamic environment usually utilizing wireless devices is limited. In this environment DDS gives satisfactory results allowing exchanging information among mission participants in real time. However, in order for the mission to be successful operators on low and high command levels or civil units need to exchange information, sharing their situational awareness and providing the basis for synchronization of forces. The interface between the WS and DDS domains is though highly recommended.

3. WS-DDS interface

The scenario proposed in our work assumes the existence of two domains – the web service domain which covers mainly web service invocations and the DDS domain that enables transferring real-time data in the peer to peer mode.

The WS-DDS dedicated interface proof of concept is the application that enables transformation of XML data (NFFI SOAP messages) from the web service domain to the data format sent in the DDS system. The opposite direction requires transformation of the DDS data format to SOAP messages sent in the web services domain. Apart from data transformation, it provides architectural and protocol transformations.

For the purpose of the proof of concept both sides (web service domain and DDS domain) send NNFI messages (NATO Friendly Forces Information) according to the STANAG 5527 [1].

WS-DDS has a web service interface that enables calling web methods providing appropriate functions. This is a front-end to the proxy functionality that connects the web service application (run on the GlassFish application server) with the OpenSplice DDS middleware. WS-DDS Interface enables the following operations:

- authorization operations:
  - login – login to WS-DDS Interface,
  - logout – logout form WS-DDS Interface,
> functional operations
  * createTopic – creates object Topic of DDS system,
  * getTopics – lists accessible Topics from the WS-DDS Topic repository,
  * createPublisher – creates Publisher object in DDS domain,
  * createSubscriber – creates Subscriber object in the DDS domain,
  * writeData – publishes data in DDS,
  * readData – reads data from DDS (pull mode).

WS-DDS Architecture has been depicted in Fig. 3.

![WS-DDS Interface](image)

Figure 3. WS-DDS Interface Architecture

The WS-DDS proxy requires that the architecture of the DDS domain is mapped to the WS domain and vice versa. For this purpose the WS-DDS interface executes the following tasks:

- it translates SOAP messages into DDS data objects (WS \(\rightarrow\) DDS communications) and DDS data objects into SOAP messages (DDS \(\rightarrow\) WS communications),
- it processes WSDL descriptions (the NFFI service in case of the proof of concept) and creates Topic data description (when the user calls createTopic operation). Additionally, it enables setting QoS parameters for the data to be sent,
- it creates registry of Topics that were created by the WS-DDS users. The list of Topics is then available for other users by the getTopics operation. This reflects the registry (or the service discovery functionality) that exists in the WS domain.

Due to the fact that the OMG is currently working on the security extensions for the DDS, the security aspects have not been analyzed.
4. OMG DDS implementations comparison

DDS standard developed by the Open Management Group (OMG) has been implemented by at least three companies that made it available for customers, namely:

- Twin Oak – CoreDX DDS,
- RTI – RTI DDS, DDS,
- PrismTech – OpenSplice DDS.

The comparison of these three implementations has been shown in Table 2. These implementations are compliant with DDS version 1.2. They have different additional features (e.g. interfaces to databases and other software integration support) and Operating Systems support (although each of them supports the most popular OSs – Windows and Linux). However, the only open source version is offered by the Prism Tech as the DDS OpenSplice Community Edition under the LGPL Commercial licence. The other two companies offer 30-days trial versions for researches and tests.

The OpenSplice version has been selected and used for our WS-DDS implementation.

According to the industry presentations all the presented implementations are interoperable in terms of the transmissions protocol [1], however, this should be further analyzed during interoperability testing with nations that have RTI and CoreDX implementations.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>CoreDX DDS</th>
<th>RTI DDS</th>
<th>OpenSplice DDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation DDS</td>
<td>C</td>
<td>C, C++, C#</td>
<td>C/C++</td>
</tr>
<tr>
<td>STANDARD INTERFACES, LANGUAGES SUPPORT</td>
<td>C++</td>
<td>Java, Ada</td>
<td>C#</td>
</tr>
<tr>
<td></td>
<td>Java</td>
<td>RTPS wire protocol</td>
<td>Java</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JMS</td>
<td>Real-Time Specification for Java</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSDL/SOAP</td>
<td>SOAP-Connector</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REST</td>
<td>DBMS-Connector</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lightweight CORBA Component Model (CCM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sockets</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Custom via adapter interface</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>CoreDX DDS</td>
<td>RTI DDS</td>
<td>OpenSplice DDS</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>OMG DDS COM-PLIANCE</td>
<td>OMG DDS v1.2 specification, including the Data Centric Publish/Subscribe (DCPD) and the Data Local Reconstruction Layer (DLRL) profiles. Wire Protocol (RTPS) 2.1</td>
<td>DDS API 1.2 – Minimum profile – Persistence profile – Ownership profile ContentFiltered-Topic &amp; QueryCondition DDS Interoperability Wire Protocol (RTPS) 2.1 Web-enabled DDS (draft) Extensible and Dynamic Topic Types (draft)</td>
<td>OMG DDS v1.2 specification, including the Data Centric Publish/Subscribe (DCPD) and the Data Local Reconstruction Layer (DLRL) profiles. Wire Protocol (RTPS) 2.1</td>
</tr>
<tr>
<td>Tools</td>
<td>CoreDX DDS Spy CoreDX DDS Multiplexor CoreDX DDS Centralized Discovery</td>
<td>Relational databases Microsoft Excel Complex Event Processing (CEP) engines Visualization platforms Application Servers and ESBs</td>
<td>Relational databases MDE PowerTools Tuner DDS TouchStone</td>
</tr>
<tr>
<td>PLATFORMS INTEGRITY</td>
<td>Linux 2.6 Windows Solaris 10 LynxOS-SE QNX 6.4 VxWorks 5.5 VxWorks 6.6 NexusWare Android</td>
<td>INTEGRITY, Linux, SE-Linux and Embedded Linux LynxOS and LynxOS-SE Mac OS X QNX Unix – AIX and Solaris VxWorks, VxWorks 653 and VxWorks MILS Windows and Windows CE/Mobile</td>
<td>AIX Linux Solaris Windows INTEGRITY VxWorks</td>
</tr>
<tr>
<td>PROCESSOR FAMILIES</td>
<td>x86 (32bit, 64bit) i686pc sun4u ARM v5 ARM v7 MIPS PPC</td>
<td>x86 (32bit, 64bit) ARM PowerPC / Cell SPARC</td>
<td>x86 (32bit, 64bit) SPARC</td>
</tr>
<tr>
<td>LICENSING</td>
<td>Commercial</td>
<td>Commercial</td>
<td>LGPL Commercial</td>
</tr>
</tbody>
</table>
5. Tests

The WS-DDS proxy has been tested against performance of processing user requests and connecting WS and DDS domains.

The tests were mainly focused on the efficiency of the WS-DDS implementation meaning the time that the appropriate transformations from the WS domain to the DDS domain were performed.

The tests were performed on the following equipment:
- WS-DDS Interface: PC with dual core processor (2.2 GHz), 2 GB RAM, Windows XP operating system,
- WS client: laptop with dual core processor (2.4 GHz), 4 GB RAM, Windows 7 operating system,
- DDS Publisher/subscriber: laptop with dual core processor (2.0 GHz), 2 GB RAM, Windows 7 operating system.

The tests were performed in the network configuration shown in Fig. 4. Each of the computers had two network cards using which they were synchronized on the basis of the Galleon NTS-6000 GPS NTP time server. According to the producer’s offer, this NTP server can maintain an accuracy of better than 50 microseconds UTC (Universal Time Clock).

![Test environment diagram]

Figure 4. Test environment

The message sent to the DDS domain by the WS client was the NFFI Track built according to the NFFI XML Schema defined for the NFFI SIP 3 service.
It was represented as the TrackMessage in the DDS domain and received by the DDS subscriber. In the opposite direction the TrackMessage was transferred by the WS-DDS into the NFFI XML message and sent to the WS client.

The experiment was carried out in two scenarios:
- Scenario 1: Efficiency tests of the WS-DDS interface invoked by one client,
- Scenario 2: Efficiency tests of the WS-DDS interface invoked by multiple clients.

In the experiment, during the WS-DDS efficiency tests, the measured time periods were:
- TP1 – WS-DDS interface processing time for the following methods: createTopic, createPublisher, createSubscriber, writeData,
- TP2 – WS-DDS interface processing time for the readData method,

The tests for every scenario were performed 30 times. This article includes the average values.

6. Experiment results

The aim of scenario 1 was to define the performance of the implemented WS-DDS proxy model measured with the time it takes to process user requests sent form the WS domain and make appropriate actions in the DDS domain. Table 3. shows results of the processing times when one user issued requests to the WS-DDS proxy. Processing times were measured in Java code at the WS-DDS proxy.

<table>
<thead>
<tr>
<th>WS-DDS proxy method</th>
<th>Processing time TP [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>createTopic</td>
<td>TP₁ = 15</td>
</tr>
<tr>
<td>createPublisher</td>
<td>TP₁ = 16</td>
</tr>
<tr>
<td>createSubscriber</td>
<td>TP₂ = 16</td>
</tr>
<tr>
<td>writeData</td>
<td>TP₂ = 16</td>
</tr>
<tr>
<td>readData</td>
<td>TP₃ = 42</td>
</tr>
</tbody>
</table>

The experiment was carried out in scenario 2 to verify the response times measured at the WS client side when multiple clients invoke WS-DDS proxy. Table 4. shows results of the WS-DDS response times in scenario 2.
Table 4. WS-DDS response time when multiple users are performing requests

<table>
<thead>
<tr>
<th>Number of clients</th>
<th>createPublisher $T_R$ [ms]</th>
<th>createSubscriber $T_R$ [ms]</th>
<th>createTopic $T_R$ [ms]</th>
<th>writeData $T_R$ [ms]</th>
<th>readData $T_R$ [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>34</td>
<td>78</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>17</td>
<td>18</td>
<td>43</td>
<td>84</td>
</tr>
<tr>
<td>50</td>
<td>43</td>
<td>45</td>
<td>46</td>
<td>67</td>
<td>178</td>
</tr>
<tr>
<td>100</td>
<td>127</td>
<td>124</td>
<td>125</td>
<td>168</td>
<td>314</td>
</tr>
</tbody>
</table>

In Figure 5, the graph depicting the WS-DDS response times for multiple users has been shown. It is visible that the response time grows exponentially when the number of users grows. This trend is true for all of the operations. However, it is visible that the writeData operation takes about 20 ms (for 1, 10 and 50 users) and 40 ms (for 100 users) more than the createTopic, createPublisher, createSubscriber methods. It results from the fact that it requires the message from WS domain to be transcoded into the DDS format and posted into the appropriate topic. The readData operation is the most time consuming one. It is caused by the fact that reading the data has been implemented without the “history option” in DDS. This means that after having received the request the WS-DDS captures data in the DDS domain for 20 ms. Then it sends the captured data to the WS domain.

Nevertheless, it must be emphasized that the Response Time of almost 320 ms for 100 clients is not very big and satisfactory when scalability is taken into account.
7. Conclusions

The article presents the concept and implementation of the WS-DDS proxy that has been implemented as the web service and enables – after being invoked by the WS client – transferring data from the WS domain to the DDS domain (writeData operation) and – getting data published in the DDS domain (readData operation). It includes the background of the cooperation of the WS domain with real-time publish/subscribe DDS middleware.

The presented proxy is a proof of concept implementation. It has been tested against performance in terms of information processing when transferring data between domains. The results proved that it adds small time overhead and is scalable.

However, the presented implementation does not provide the functionality of using the HISTORY option in the DDS middleware. This option gives the middleware the possibility of storing pieces of information that have been published in particular topic. It would be beneficial in the WS-DDS proxy to define x messages that would be stored in DDS and, after the WS client invokes the readData method, these messages would be immediately taken from DDS, transformed to the WS domain message format and sent to the client. It is envisaged to provide this functionality in the nearest future.

Providing the possibility of connecting the DDS and WS domains remains an interesting functionality in the operational context of civil – military cooperation. The WS domain can be seen as the civil domain looking for information about the requests for help in the civil community on the area military forces operation. The DDS domain can be the domain of military unit working on the field. The DDS publishers can send reports on the area they are in that can be published then using the WS-DDS proxy in the WS domain (sent to the WS consumers or delivered to the web service provider to be sent further along). Similarly, the information from the WS domain (e.g. notifications about the local situation) should be forwarded to the DDS subscribers. WS-DDS proxy can strongly support these actions and bring significant benefits in operational context.

REFERENCES


[12] WSDL standard main page, http://www.w3.org/TR/wsd1
