ORTE: The Open Real Time Ethernet

Petr Smolik, Pavel Pisa

Czech Technical University in Prague

Department of Control Engineering – petr.smolik@wo.cz, pisa@cmp.felk.cvut.cz

Abstract: This paper describes the development process of the ORTE, a publish/subscribe communication middleware. The ORTE implements RTPS (Real Time Publish Subscribe) communication architecture conforming OMG specification. RTPS architecture makes design, development and deployment of distributed applications much easier than traditional architectures. It removes one-to-many communications challenges at the core of data-centric applications and distributes messages with low latency without any single point of failure. The RTPS architecture is the ideal communication middleware for distributed systems.

Keywords: middleware, publish-subscribe, DDS, RTPS

1 INTRODUCTION

In the ORTE project we are developing communication middleware confirming the OMG Data Distribution Service (DDS) for Real-Time Systems specification [9] (OMG Document formal/07-01-01). The specification defines a service for efficiently distributing application data between participants in a distributed application.

The recently-adopted DDS specification is divided into two components. The first defines an Application Level Interface (API) and behavior of the DDS that supports Data-Centric Publish-Subscribe (DCPS) in real-time systems. Second optional Data Local Reconstruction Layer (DLRL) allows distributed data to be shared by local objects located remotely from each other as if the data were local. The DLRL is built on the top of the DCPS layer.

The DDS specification also includes a platform specific mapping to IDL and therefore an application using DDS is able to switch among DDS implementations with only a re-compilation. Therefore DDS addresses the 'application portability.'

The specification does not address a protocol used by the implementation to exchange messages over transport layer such as TCP/UDP/IP, so different implementations of DDS will not interoperate with each other unless vendor-specific “bridges” are provided.

One of the transport protocol suitable for DDS communication is Real-Time Publish Subscribe (RTPS) wire protocol defined by OMG [10] (OMG document formal/06-08-02). The RTPS was specifically developed to support the unique requirements of data-distributions systems. It is a field proven technology that is deployed in thousands of industrial devices.

The rest of this paper is structured as follows. Section 2 provides background informations of the RTPS communication architecture and related works. Section 3 and 4 describe our implementation of the ORTE in terms of core, internal implementation and shows some code examples. Last section presents concluding remarks.

2 BACKGROUND AND RELATED WORKS

2.1 Publish-subscribe

The publish-subscribe (PS) architecture is designed to simplify one-to-many data-distribution requirements. In this model, an application “publishes” data and “subscribes” for data reception. Publishers simply send data anonymously, they do not need any knowledge of the number or network location of subscribers. Similarly subscribers simply receive data anonymously, they do not need to have any knowledge of the number or network location of the publisher.

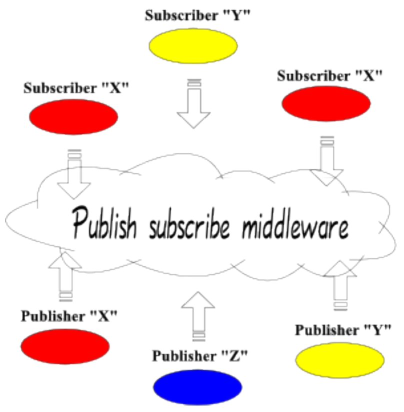

Fig.1 Publish-subscribe messaging

Each message in a publish/subscribe environment has an associated topic, message type or any filtered criteria, which is used to couple data publishers and subscribers. Each participant dynamically registers itself in the environment, where other peer modules are active. By specifying the topics it wishes to publish and subscribe, the connection is automatically created and data transfer can begin. This coupling is illustrated in Figure 1, where publish/subscribe peers are connected through a middleware “cloud”, which provides an abstraction for the network environment and communication.

Publish/subscribe architecture can easily emulate traditional forms of the communication such as point-to-point (1 publisher, 1 subscriber), broadcast (1xN, MxN) or client/server (Nx1). For example scenario MxN, different publishers can declare the same publication so that multiple subscribers can get the same issues from multiple sources.

The basic communication model of PS is unidirectional data exchange. For bi directional communication is necessary to create two pairs of publishers and subscribers. PS is much more efficient than client-server design like CORBA, because of no request traffic and the direct data transfers makes. The architecture is naturally event-driven.

On other hand, publish/subscribe architectures are not good at sporadic request/response traffic, such as is for example file transfer. For this purpose, protocols like FTP or HTTP are more efficient.

2.2 *Real Time Publish Subscribe*

Publish/subscribe distribution is gaining popularity in many distributed applications, such as robotics, industrial automation and embedded systems. Notably leading to industry standard as the Real-Time Publish-Subscribe (RTPS) network protocol [13] (IDA 2001) and Data Distribution Services (OMG 2006). Implementations of RTPS can be found in industrial commercial products such as Network Data Delivery Serves (NDDS) from Real Time Innovations (RTI) and Thales SPLICE DDS from Thales Naval Netherlands and Prism Technologies. NDDS implements the DCPS fully and the DLRL partially. The RTPS protocol is also available in the Programmable Logic Controllers (PLC) from Schneider Electronic.

Publish/subscribe offers clear advantages for real-time applications. RTPS adds publication and subscription timing parameters and properties so the developer can control the different types of data flow and achieve their application's performance and reliability goals.

RTPS has low bandwidth and lightweight process requirements. Was specifically designed to work on top of unreliable and connection-less network transport such as UDP/IP.

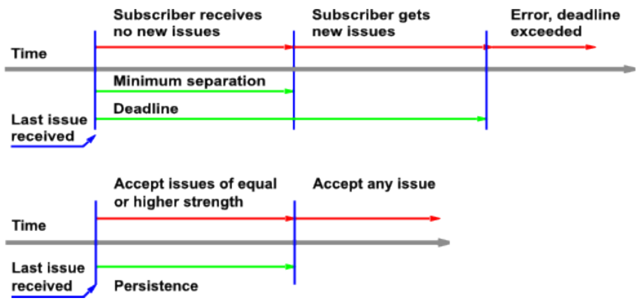

Fig.2 Publisher and Subscriber timing and arbitration QOS in message delivery

Each publication is mainly characterized by four parameters: topic, type, strength and persistence. The topic is the label that identifies each data flow. The type describes the data format. The strength indicates a publisher's weight relative to other publishers of the same topic. The persistence indicates how long each publication issue is valid. Figure 2 illustrates how a subscriber arbitrates among publications using the strength and persistence properties.

When there are multiple publishers sending the same publication, the subscriber accepts the issue if its strength is greater than the last-received issue or if the last issue's persistence has expired.

Similarly subscriptions are identified by four parameters: topic, type, minimum separation and deadline. The topic is the label that identifies the data flow, and type describes the data format (same as the publication properties). The relationship between a publication is created under this definition: topic, type. Minimum separation defines a period during which no new issues are accepted for that subscription. The deadline specifies how long the subscriber is willing to wait for the next issue. Figure 2 illustrates the use of these parameters. The minimum separation protects a slow subscriber against publishers that are publishing too fast. The deadline provides a guaranteed wait time that can be used to take appropriate action in case of communication delays.

There are sever ways, how to optimize communication data throughput in an implementation. One of the usable method is multicast. This method dramatically reduce the number of the messages exchanged in the network. Another way, how to increase throughput is tunneling local data traffic for instance through shared memory. The RTPS specification keep all this possibilities.

In the open source world, exists tool for analysis and capturing RTPS (IDA 2001) data traffic on the network. In standard distribution of Wireshark have been designed a interpreter of RTPS messages [11]. Wireshark is the world's foremost network protocol analyzer, and is the de facto (and often de jure) standard across many industries and educational institutions.

2.3 *DDS*

DDS and RTPS share same origin and are complementary from application interface. Difference is only at the definition of wire protocol. DDS keeps open wire protocol to allow different vendors specific implementations. RTPS (OMG, 06-08-02) is one possible standard implementation of DDS wire protocol.

DDS introduces a virtual *Global Data Space* (Figure 3) where applications can share information by simply reading and writing data-objects addressed by means of an application-defined name (**Topic**) and a **key**. DDS provides fine and extensive control of *QoS* parameters, including *reliability*, *bandwidth*, *delivery deadlines*, and *resource limits*. DDS also supports the construction of local object models on top of the *Global Data Space*. This reconstruction is done by optional Data Local Reconstruction Layer (DLRL).

DLRL is not only a possible extension of DCPS. For instance exists project SQLbusRT which implements distributed database over DCPS. The objectives of SQLbus/RT is to create Real Time Publish Subscribe SQL. Data storage of SQLBusRT is provided by MySQL database engine.

All interfaces (DCPS and DLRL) are described in the specification by using Independent Definition Language

(IDL). IDL helps create platform and system independence.

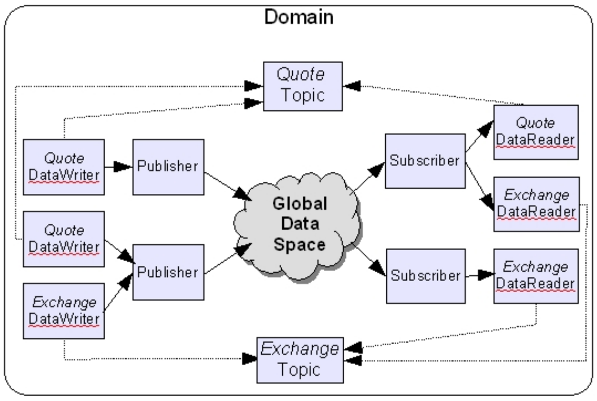

Fig. 3 DCPS concept

The DDS relation with the CORBA is that it only specifies CORBA IDL interfaces for the setup, control, and configuration of the application and assumes that the data transmission occurs via mechanisms other than CORBA. This enables DDS implementations to achieve higher performance and better quality of service than the CORBA-based alternatives.

2.4 *Related works*

Except commercial versions of DDS, there exists another open source implementations. Object Computing, Inc (OCI) supports one of them. Their implementation is called OpenDDS [5]. OpenDDS supports the capabilities defined in the DDS 1.0 Specification and implements "Minimum profile" from them. The middleware is written in C++ language.

The architecture of the OpenDDS takes advantages from he Adaptive Communication Environment (ACE) to provide a cross platform portability. OpenDDS also leverages capabilities of TAO (The ACE Orb), such as its IDL compiler and as the basis of the OpenDDS DCPS Information Repository.

Transport protocol is separated from the higher level protocols by means of an extensible transport framework (ETF). The Figure 4 below shows the ETF aspects of the architecture.

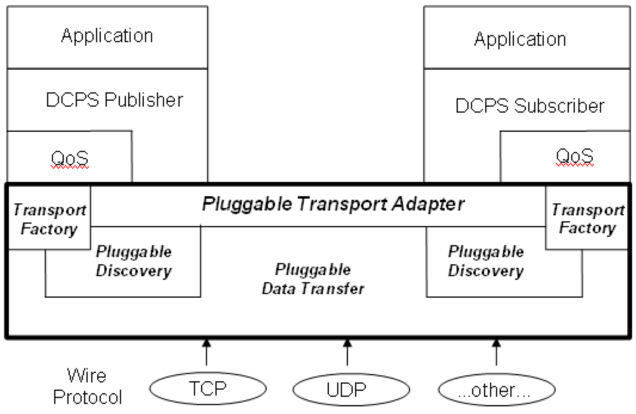

Fig.4 OpenDDS architecture

OpenDDS ETF is not standardized. Data are usually sending through TCP/IP communication protocol, UDP or user can write a self implementation. A similar approach is used within TAO and is called "Pluggable Protocols".

Although ETF is well designed to be scalable, not all kind of communication protocol can be solved by this model. We discussed about possibility of integration RTPS like a new ETF at the beginning . Unfortunately, the implementation of RTPS protocol needs more than just self implementation ETF. Communication messages are marshaled/demarshalled directly in OpenDDS by specific way. RTPS implementation based on OpenDDS can not be done without the change of architecture and changing many parts of code in upper layers of middleware. Also this kind of transportable adapter is not suitable for a RTPS implementation.

Globally, OpenDDS offers to developers standardized application interface with a proprietary version of communication protocol.

Another an open source version of DDS is JacORB DDS [2]. The distribution provides an open-source Java-based DDS-DCPS implementation. The purpose of this implementation is only pedagogical and demonstrations. The communication is working over CORBA. The last version has be released in October 2005 and from this date was not updated.

3 ORTE – THE WORKING IMPLENTATION

The open source implementation of the RTPS protocol has been developed at the Czech Technical University in Prague as one of the results of the OCERA project. At the beginning of OCERA project was accessible only IDA RTPS specification. It is reason why we designed to develop ORTE under this specification. First version of the DDS specification have been introduced at the end of project OCERA. In that time it was not possible to change API to DDS and the ORTE remained with proprietary API version. The IDA RTPS specification does not cover all OMG RTPS (document/06-08-02) messages. New specification extends protocol with the partial backward compatibility.

Although the object concept of RTPS would be ideal for an object oriented programming language such as C++, the final ORTE implementation is done in C language, since it allows simple porting of ORTE to different operating systems, mainly those with the real-time behavior. Figure 5 and Figure 6 show how simply ORTE can be used.

```
ORTEPublication *p;
NtpTime persistence, delay;
ORTEInit();
d = ORTEDomainAppCreate(ORTE_DEFAUL_DOMAIN,
                NULL, NULL, ORTE_FALSE);
ORTETypeRegisterAdd(d,"HelloMsg",NULL,NULL,64);
NTPTIME_BUILD(persistence, 3);//is valid for 3s
NTPTIME_DELAY(delay, 1);
p = ORTEPublicationCreate(
        d,      // pointer to application object
        "Example HelloMsg",  // name of topic
        "HelloMsg",          // data type description
        &instance2Send,      // output buffer
        &persistence,   // persistence of publication
        1,      // strength of publication
        sendCallBack,        //pointer   to   callback
function
        NULL,   //user parameters for callback
        &delay);// period for timer, callback
```

Fig.5 The skeleton of the ORTE publisher

The subscribing application needs to create a subscription with publication's Topic and Type. A callback function is called whenever a new publication from the publisher is received.

```
ORTESubscription *s;
NtpTime deadline, minimumSeparation;
ORTEInit();
d=ORTEDomainAppCreate(ORTE_DEFAUL_DOMAIN,
                    NULL, NULL, ORTE_FALSE);
ORTETypeRegisterAdd(d,"HelloMsg",NULL,NULL,64);
NTPTIME_BUILD(deadline, 20);
NTPTIME_DELAY(minimumSeparation, 0);
p = ORTESubscriptionCreate(
           d,        // pointer to application object
           IMMEDIATE,        // mode of subscription
           BEST_EFFORTS,     // type of subcsription
           "Example HelloMsg",// name of topic
           "HelloMsg",       // name of data type
           &instance2Recv,   // pointer to output buffer
           &deadline,        // deadline
           &minimumSeparation,// minimum separation
           recvCallBack,     // callback function
           NULL);            //user parameters
```

Fig.5 The skeleton of the ORTE subscriber

The ORTE was tested across different platform such as Linux, Windows, MacOS and FreeBSD running on littleendian as well as bigendian architectures. Serialization and deserialization support is done by IDL compiler, which is based on ORBIT IDL compiler from the GNOME project. Java programming interface is accessible through Java Native Interface (JNI). The RTPSEye (Pokorny, 2005) is Java based program used for online browsing of RTPS communication object's parameters.

ORTE source package contains a graphical demonstration program. This demonstration application is a self-contained introduction to the elegance and power of publish-subscribe networking implemented in the ORTE.

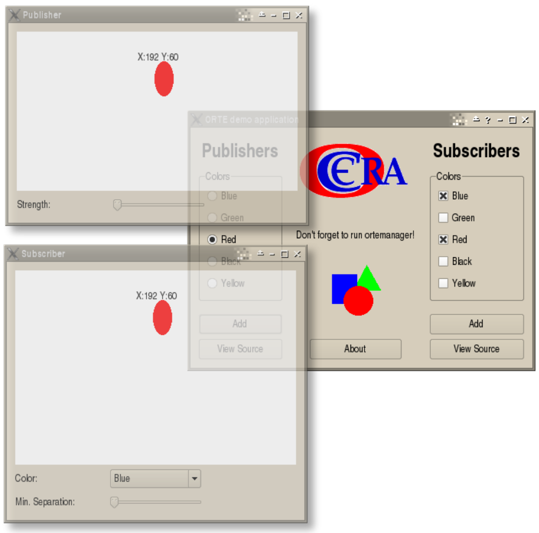

Fig. 6 ORTE demo application

The finial ORTE version (0.3.2) is fully complaint with IDA RTPS standard. The communication interoperability was successfully tested with NDDS version 3 from RTI and Schneider Unity Premium PLC – TXSP571634M. Although the PLC supports RTPS communication, the possibilities of data exchanges are limited. In the tested configuration were transmitted variables with fixed length 4 bytes (unsigned long).

The ORTE have been used like basic communication network in some real projects. Imtech ICT TS in the Netherlands have adopted the ORTE implementation, in large and interesting projects. One of them is aerodynamic wind tunnel, where the ORTE is used like core message exchange mechanism. The project Seaware [4] implements a PS middleware for networked vehicle systems with the ORTE. Next one is the Eurobot2008 (Sojka, CTU), which also takes advantages of RTPS architecture.

4 ORTE – DEVELOPMENT VERSION

In the next phase we develop the ORTE according OMG RTPS specification (OMG document formal/06-08-02) with the respect of DDS API. The ORTE version 0.3 implements most of things from this specification. To be complaint with the DDS RTPS, the ORTE needs totally revise API and extends internal message sending mechanism and add new telegrams. The revision needs to change some of the architecture concept. Responses from projects using ORTE are indicated to native support of object interface for API. Currently was done by making a wrapper class. Big projects are usually written in C++. DCPS API is also naturally mapped into C++, thought the fact of object inheritance. With the future point of view, we decided totally rewrite ORTE by using C++ language. We would like to provide a new ORTE version in the project FRESCOR.

The inspiration for new development have been taken from project OpenDDS. Platform independence was solved same way by using ACE components. ACE provides a rich set of reusable C++ wrappers and framework components, that perform common communication tasks across a range of O/S platforms. Such communication tasks provided by ACE include event demultiplexing, and event handler dispatching, signal handling, service initialization, interprocess communication, shared memory management, message routing, dynamic reconfiguration of distributed services, concurrent execution, synchronization and support for Common Data Representation (CDR), which was introduced in CORBA and is adapted by DDS. Including TAO dependency like in OpenDDS takes very huge footprint for small target platforms and the benefits from this dependency are not so significant.

First step in the ORTE development which has to be solve is concerning serialization/deserialization communications objects. The object description is written in IDL files. Parsing IDL files is done by library libIDL originally created for the GNOME project. LibIDL is a small library creating parse trees of CORBA v2.2 compliant IDL files. The skeleton for writing ORTE IDL compiler (orte-idl) was used IDL compiler from the ORBIT project. Both compilers take advantages from libIDL.

Basic data types (short, long, float, double, char, boolean,octet) in IDL statements are mapped on basic C++ types. Complex data types are mapped on types defined in Table 1.

OMG IDL	**C++**
`struct`	`C++ struct`
`enum`	`C++ enum`

string	char *
wstring	DDS::StringManager *
sequence	C++ class
array	C++ array

Tab. 1 Mapping complex IDL types on C++ types

If in the IDL are used types with unbounded size, the mapping problem is dramatically increased. The ORTE IDL version 0.3 have limitation on using bounded types. In new version of ORTE this limitation has been solved. For example, if a unbounded type (sequence, string, array) is used in a struct, than different mapping have to be used for serialization/deserialization. The compiler has to understand all basic and complex types and has to know correct handling with this types on different compilation level.

The result of the orte-idl is set of files. The IDL development process is illustrated by Figure 7.

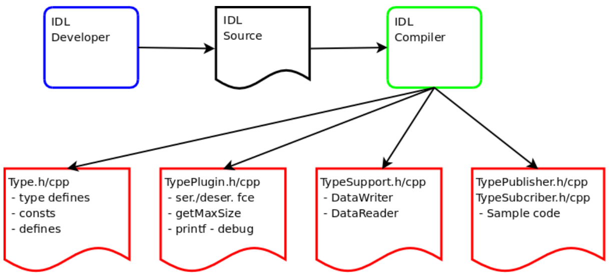
IDL Developer
IDL Source
IDL Compiler
Type.h/cpp - type defines - consts - defines
TypePlugin.h/cpp - ser./deser. fce - getMaxSize - printf - debug
TypeSupport.h/cpp - DataWriter - DataReader
TypePublisher.h/cpp TypeSubcriber.h/cpp - Sample code

Fig. 7 ORTE IDL development process

The orte-idl currently generates from the file Type.h files Type.h/cpp and TypePluggin.h/cpp and correct maps bounded/unbounded data type to DDS types.

IDL interfaces are directly mapped into C++ function definitions. DDS specification is defined in IDL file. The orte-idl compiler is used for compilation of the DDS IDL file to produce skeleton for writing ORTE DDS implementation.

The ORTE architecture is splited on server parts. Ortecore contains definition of basic and complex DDS types. There is also support for correct handling with serialization/deserialization this types. Ortecore is almost done. Next layers of the ORTE architecture (PIM, PSM, Transport) needs to be implemented. Many inspiration for this development could be gain from the OpenDDS.

5 CONCLUSION

This paper has described the ORTE (Open Real-Time Ethernet), an open-source implementation of the RTPS standard. The authors see RTPS based implementation as promising middleware for real-time network communications.

Although there exists open source implementation of the DDS specification called OpenDDS, the wire protocol of this implementation is not confirming to RTPS specification. Can be integrated a new transport protocol by using extensible transport framework (ETF), but not a whole new communication protocol.

The ORTE version 0.3 implements RTPS under IDA specification (2001) and is full ready to be used in serious projects. The interoperability was successfully tested against the commercial product from Real Time Innovations' NDDS3.

Ongoing ORTE implementation will includes RTPS specifications under OMG document (formal/06-08-02) written in C++. The platform independence will be solved by ACE components. In the recent version is implemented IDL compiler and core function for serialization and deserialization IDL basic and complex types.

6 ACKNOWLEDGEMENT

This work was supported by the project IST 35102 – OCERA and IST 5-034026 – FRESCOR. Finally, thanks are also to all people who helps with the ORTE development.

7 REFERENCES

[1] ACE Adaptive Communication Environment, http://www.cs.wustl.edu/~schmidt/ACE.html
[2] DDS JacORB, open source DDS in Java http://www.dele.imag.fr/users/Didier.Donsez/dev/dds/readme.html
[3] Dolejš, O., Smolík, P., Hanzálek, Z.: On the Ethernet Use for Real-TIme Applications, 5th IEEE International Workshop on Factory Communication Systems, WFCS, Vienna, September 22-24, 2004.
[4] Marques, E. R. B., G. M. Gonçalves, J. B. Sousa (2006). Seaware: A Publish/Subscribe Middleware for Networked Vehicle Systems. Robotica 2006.
[5] OpenDDS, open source DDS over ACE http://www.opendds.org
[6] ORTE open-source code - CVS, http://cvs.sourceforge.net/viewcvs.py/ocera/ocera/components/comm/eth/orte/
[7] ORTE documentation, http://www.ocera.org/archive/ctu/public/components/ethdev/ethdev-0.1.tgz
[8] OCI, Object Computing Inc (n.d.). TAO DDS. http://www.ociweb.com/products/dds
[9] OMG, Object Management Group. Data Distribution Service for Real-time Systems Specification, v1.2, 2007 http://www.omg.org/technology/documents/formal/data_distribution.htm
[10] OMG, Object Management Group (2006). The Real-time Publish-Subscribe Wire Protocol http://www.omg.org/cgi-bin/doc?ptc/2006-08-02
[11] Pokorný, L. Support tools for RTPS communication ČVUT 2005
[12] RTI, Real Time Innovation Inc, RTI DDS http://www.rti.com/products/data_distribution/index.html
[13] RTI, Real Time Innovation Inc, Real-Time Publish-Subscribe Wire Protocol Specification, Protocol Version 1.0, Draft doc, version 1.17. http://www.rti.com/products/ndds/literature.html, 2001
[14] TAO, CORBA implementation http://www.cs.wustl.edu/~schmidt/TAO.html