# Application of a CAN BUS transport for DDS Middleware

Rojdi REKIK Salem HASNAOUI
*Communication Systems Research Laboratory - SYSCOM*
*National School of Engineering of Tunis-ENIT*
*Rekik.Rojdi@topnet.tn Salem.Hasnaoui@enit.rnu.tn*

## Abstract

*The Publish/Subscribe paradigm matches well with these systems. Data Distribution Service (DDS) is a Publish/Subscribe data-centric middleware. It specifies an API designed for enabling real-time data distribution and is well suited for such complex distributed systems and QoS-enabled applications. Unfortunately, the need to transmit a large number of sensor measurements over a network negatively affects the timing parameters of the control loops.*
*The CAN-bus enables the information from a large number of sensor measurements to be conveyed within a few messages. Its priority-based medium access control is used to select the sensor messages with high timing constraints. This approach greatly reduces the time for obtaining a snapshot of the environment state and therefore supports the real-time requirements of feedback control loops.*
*The use of the " Publish/Subscribe/Distribute " paradigm and the underlying real -time CAN-Bus is a currently research topic an d only today few works exist in this area. These activities are headed by University of ULM & German National Research Center for Information Technology and by Software Architecture Lab, Seoul National University as well as by our "Control and Communication Technologies" research group at the National School of Engineering of Tunis (ENIT), Department of Computer and Communication Technologies, headed by myself.*
*The main objective of this paper is to demonstrate how DDS API is implemented on a CAN-Bus.*

## 1. Introduction

Today's embedded software applications are increasingly distributed; they communicate data between many computing nodes in a networked system. This includes applications in aerospace, defense, distributed simulation, industrial automation, distributed control, robotics, telecom equipments and networked consumer electronics.
In the current days, the industry is challenged by the demand for productivity, quality, safety, environmental protection and the increasing degree of horizontal and vertical integration. However, large and complex Distributed Control systems (DCS) cannot be efficiently and safely managed without providing more powerful inter-object communication patterns. The need for interoperability of control functions on different hierarchical levels covering networks, middleware architectures and application objects is the real challenge for system integrators when adapting and connecting components available from various suppliers. For many years intelligent control systems have been the focus of international standardization organizations, industrial consortia and research groups providing more comprehensive data models and reference architectures. These efforts succeeded to the elaboration of standards which map well with the called "Embedded Communications Challenge". DAIS, OPC-DX, DDS and SWE are the most recent and powerful standards for intelligent control systems.

## 2. Related Works

Real-time data distribution has recently emerged as an important area of research. There was a workshop dedicated to the topic (The First Workshop on Data Distribution for Real-Time Systems [4]) in May of 2003. The Object Management Group (OMG) contributes to the research efforts by standardizing data distribution in a middleware service. In [5], the problem of scheduling the broadcast of real-time data is considered. It provides an approximate version of the Longest Wait First heuristic that reduces overhead. Similar work [6] describes a Broadcast on Demand technique that schedules the broadcast using earliest deadline first, periodic or hybrid scheduling algorithms. The work described in [7] is a speculative data dissemination service that uses geographic and temporal locality of reference to determine which data to be disseminated. These techniques take into account the deadline timing constraints of the clients, but consider neither the data temporal consistency nor the use of underlying real-time networks. An application area that has provided various research efforts towards data distribution is embedded sensor networks [8, 9, 10, 11, and 12]. While all of the work described here provides valuable insights into solving the problem of data distribution in sensor networks, none considers real-time characteristics of the data or of the applications. That is, neither deadlines on data delivery nor temporal consistency of the data is supported.

A large amount of real-time data distribution research has been done at the University of Virginia (UVa) in the context of wireless sensor networks [13,

978-1-4244-4457-1/09/$25.00 ©2009 IEEE

14, 15, and 16]. This work does address the deadlines of requests. Also, temporal validity is considered in the sense that data values are reported before they expire, but with corresponding confidence values. However, it does not provide assurance that the data is temporally valid when it arrives at the requestor. PrismTech[17] has a product called OpenSplice[18] which is compliant with real-time networking. DDS implementation compliant with CAN-based networks have not been treated yet, but similar works can be mentioned such as ROFES[19]. In the context of ROFES platform, S. Lankes, A. Jabs and T. Bemmel describe the implementation of a CAN-based connection-oriented point-to-point communication model and its integration into Real-Time CORBA; but this project hadn't been extended to support data distribution service.

These research works has been enforced by several commercial products which are working on becoming compliant with the OMG's Data Distribution specification.

Real-Time Innovations [20] has a product called NDDS that provides publish-subscribe architecture for time-critical delivery of data. Thales Naval Nederland [21] has a product called SPLICE [22] that provides a data-centric architecture for mission-critical applications. Both of these products provide valuable real-time features in data distribution. But neither guarantees data temporal deadlines nor real-time network support.

## 3. Real-time and Embedded systems

DDS is not the only concept that can be adopted for Distributed Real-time and Embedded systems (DRE). The automation and control components are just an example of distributed components. The most common and powerful patterns are [23]:

- Event-Driven Data Distribution
- Request–Reply services
- Content-Based Event Notifications
- Continuous Data Distribution

### 3.1 Event-Driven Data Distribution

Event-Driven Data Distribution (EDDD) is used when an application is controlled via events and therefore when an event is lost, the execution of an application is stopped. In EDDD the transmitted message is an event instead of signal value. The mechanism should guarantee the delivery of every event to all receivers. The CORBA Messaging specification supporting the Asynchronous Messaging Invocation (AMI) model maps well with this pattern.

### 3.2 Request–Reply Service

Request–Reply Service (RRS) is the easier way to obtain data or to start a service. Reply can be synchronous or asynchronous. In the later case, a call-back function is invoked when the reply is received. RRS is provided by all distributed environments and in particular CORBA architecture.

### 3.3 Content-Based Event Notification

Content-Based Event Notification (CBEN) is used to notify changes in the state of components. Alarms and other notification messages are sent to all interested receivers or stored in a history database. (A message is called alarm when it forces the receiver to stop its normal operation for executing a special action.) Event producers issue event notification messages to a system-wide common space (Mailboxes or Channels). Event consumers, in turn, subscribe to specific types of events by defining an event filter. By comparison to EDDD, CBEN represent a separate domain of events that are not involved in controlling the application's normal execution. However, reliability of alarms and events are also required. CBEN can be achieved by CORBA Event Service (no filtering) or by CORBA Notification Service (with filtering). OPC, DAIS and Sensor Web Enablement (SWE) are DCS that are ranged within this category of CBEN patterns.

### 3.4 Continuous Data Distribution

Continuous Data Distribution (CDD) is used for cyclical data transfer (or when a signal value exceeds an application specific threshold). It permits a producer to transmit a fresh data value frequently. The OMG's DDS [5] and OPC-DX [6] support CDD as well as RRS and CBEN. In DDS, publishers give data to middleware. The middleware takes care of the distribution and notifies the subscriber when the message arrives.

In case 3.1, the name or location and the types of events are required at compile time. In case 3.2, the name or location of the service can be obtained via a naming or a directory service. In case 3.3, producers and consumers are de-coupled. Alarms and Events are sent to a channel; the consumers retrieve them from the channel by applying to them filters with some criteria. In case 3.4, the only property a producer (publisher) needs to communicate with a consumer (subscriber) is the name and the definition of the data (data-centric). Signals, alarms, notifications and events are identified with unique topic names, such as "pressure" or "temperature". The publisher does not need any information about the subscribers, and vice versa.

CDD and CBEN use, respectively, the Data and Service contents instead of the identity of the producer and therefore called Content-Based Publish/Subscribe. The loose coupling of Publishers and Subscribers, in CDD and CBEN, enables a Publish/Subscribe-based architecture to be scalable as new publishers and subscribers can be seamlessly integrated into the existing application without affecting existing code and hence, several redundant producers can run in parallel. Discovery is the common approach permitting consumers to discover most suitable data sources and services on the basis of their properties. So, the Publish/Discover/Subscribe/Distribute pattern covers a global vision of the new Content-based communication model which resolves the embedded communications challenge. DDS targets real-time systems; The DDS specification is less explicit about the scheduling mechanisms that should be used to coordinate these policies and to make best benefit when exploiting the underlying facilities of the real-time network, in occurrence the CAN-Bus.

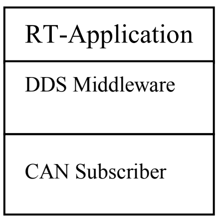
## 4. DDS and QoS parameters

Each QoS policy may have multiple parameters associated with it, such as the data topic of interest, data filter criteria, etc. Each parameter can also be assigned one of a range of values, a range of integers for the maximum number of data messages stored for transmission, or the set of regular expressions used as filtering criteria. Not all combinations of QoS policies/parameters deliver the required system QoS and it can be tedious and error-prone to manually transform a valid QoS policy configuration design to its implementation in a particular middleware platform. An efficient DDS implementation must permit:

- Association between entities and QoS policies
- Managing QoS Policy Configuration Variability.
- Checking compatibility and consistency constraints.
- Ensuring QoS compatibility
- Ensuring QoS consistency
- Ensuring Correct QoS Implementation

## 5. The Proposed Simulator

The framework architecture for the simulator is a set of nodes connected via DDS middleware and CAN drivers when just the network controllers are replaced by pipes between a publisher represented by a CAN Dispatcher and subscribers.




The communication between nodes is achieved due to publish subscribe interface via the Global Data Space that is represented a file which is the same on the all nodes. The middleware has to keep track of the data objects instances. Each data object is identified by the combination of a topic object, its Data Writer and Value.

### 5.1 Transport Priority computation and Sending Frames in the CAN_Bus

The main idea is to define a struct that integrates sorted topics by theirs deadlines (or priorities) and for each topic its Datawriters list, sorted also by deadline. This struct is just a TreeMap. The topic is its key and correspondent value is the DataWriter list.

| Key | value |
|---|---|
| TopicObject_1 (Higher priority) | ListDataWriter_1 (sorted by deadline) |
| ⋮ | |
| TopicObject_n (Lowest priority) | ListDataWriter_n (sorted by deadline) |

Figure 2. TreeMap Structure

However, we must add a sort criterion to the topic definition which is inexistent in the DDS specification. The method added is called "compareTo(Object o)" which returns an integer. The figure 3 depicts this addition.

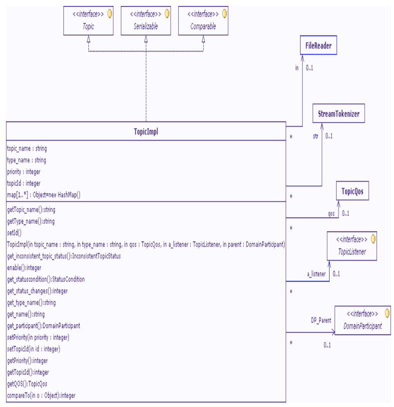



Figure 3. New definition of TopicImpl Class

The same work is done for the DataWriter. We notify that this feature must be added to DDS specification in the future versions permitting to developers to use the same designated method.

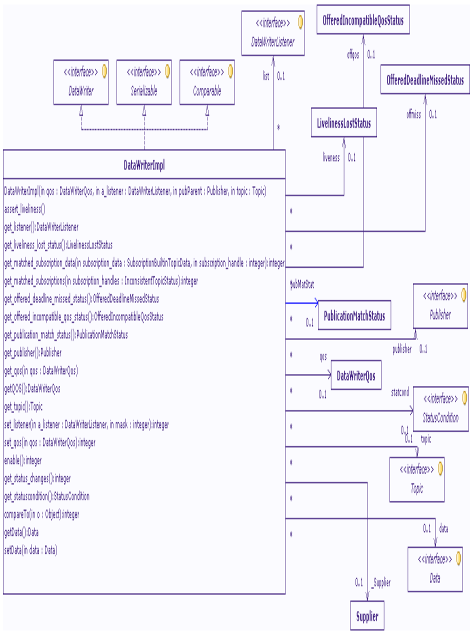


Figure 4. New definition of DataWriterImpl Class

We define a ListDataWriter class permitting to add, remove and sort the DataWriters also by deadline.

We implemented a ListTopicDataWriter class which manages the TreeMap Struct, represented in figure 2. The main operation of this class is to set the priority for the topics.

For sending in the CAN_Bus, represented by CAN drivers and use of pipes, we retrieve the necessary information from the ListTopicDataWriter, to construct the CAN data frames. To send in the CAN_Bus, we use the write (Topic topic, Object o) method from a Supplier class, just indicated in the DDS specification.

```
DataFrameCan frame =new DataFrameCan(topic.getPriority(),nodeId,topic.getTopicId(),data);
DataWriter dataW= data.getDataWriter(); Duration_t duration = dataW.getQOS().deadline.period;
Framelist.addCanFrame(frame, duration);
```

To simulate the data writing in the CAN_Bus, we collect the CAN data frames within a tree map, which is analogue to the structure represented in figure 2.

| Key | value |
|---|---|
| DataFrameCan_1 (Higher priority) | deadline.period of its DataWriter |
| ⋮ | |
| DataFrameCan_n (Lowest priority) | deadline.period of its DataWriter |

Figure 5. TreeMap Structure for sending in the CAN_Bus

We implemented a class called "ListFrameCanDeadline" to manage the above structure. The figure 6 depicts the algorithms related to the simulator implementation.

## 5.2 Receiving Frames From the CAN_Bus dependent DataReaderQoS

To read a CAN frame, we use a subscriber CAN class which retrieve the DataFrameCan list (within the pipe). The subscriber CAN class notify with an event the listeners registered for a topic the arrival of CAN data frames.

```
protected void notify(int topicId){
Event evt= new Event(this,getValue(topicId)); Vector v; synchronized(this) { v = (Vector) Listeners.clone(); }
    Iterator iter = v.iterator(); while (iter.hasNext()) { DataReaderListener dataTemp=( DataReaderListenerImpl) iter.next(); if (dataTemp.getTopicId()==topicId){
        dataTemp.change(evt,topicId);
        }
        }
```

Figure 6. Notify Method implementation

This class manages the DataReaderlisteners (add, remove). The delivery of data is done for the DataReaders which have a listener for the topic, respectively the required QoS (DataReaderQoS). The on_data_available DataReader method.

```
int [] on_data_available (ListeFrameCanDeadline list)
{ // Each CAN data frame contain 8 bytes int []val=new int [8]; Map map =null; map=list.getTopicsDataFrameCan(get_listener().getTopicId()); for (Iterator it=map.entrySet().iterator();it.hasNext();)
        { Map.Entry e=(Map.Entry)it.next();
        Duration_t dr=(Duration_t)e.getValue(); int sec=dr.sec; int nanosec=dr.nanosec;
if ((sec==qos.deadline.period.sec)&&(nanosec==qos.deadline.period.nanosec )){ val=((DataFrameCan)e.getKey()).getDatas();
            }
        }return val;
 } }
```

Figure7. Java Syntax of : on_data_available Method implementation for the CAN_Bus

The DataReaderImpl is resume in Figure 8

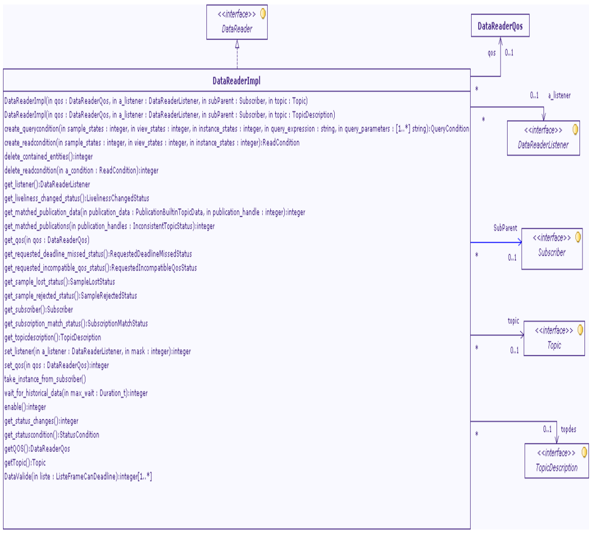


Fig. 8: DataReaderImpl class

We abstract in the following figures the algorithms and the project for sending and receiving frames from the CAN_Bus using the DDS middleware.

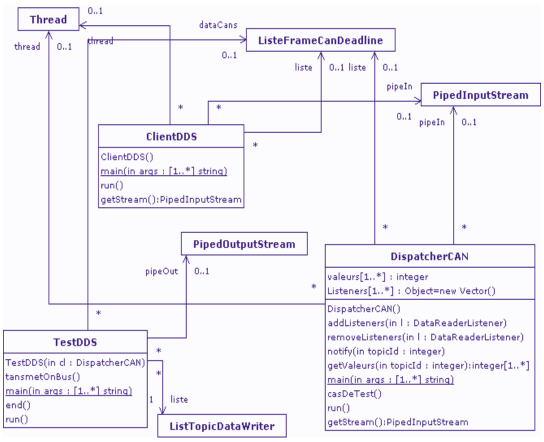



Figure 9. The application test

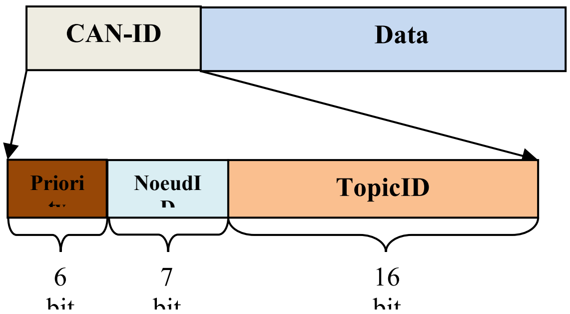


Figure 10. DataFrameCAN Format

**// registration structures**

```
Struct CANDDSID{
// Priority Max= 63
int:  Priority
// TopicIDMAX= 65535
int : TopicID
//NoueudIDMAX= 127
int : NoueudId
}
Struct DataFrameCAN
{
// 29 Bits
CANDDSID: id
Data : data
            }

List : DataFrameCANList
{
DataFrameCAN :dataCAN
Deadline:  duration
}
```

The Implementation of DataFrameCAN is demonstrated in Figure 11.

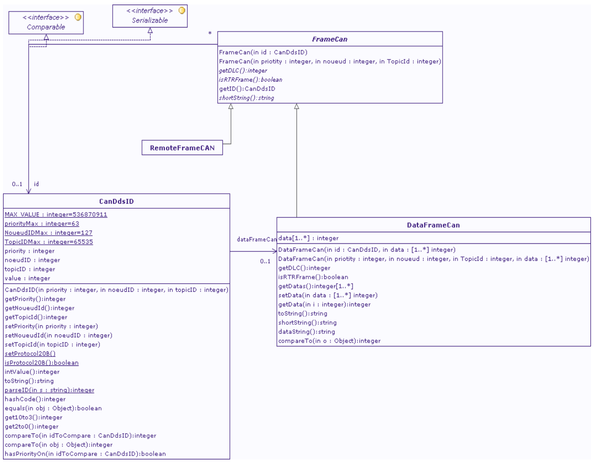



Figure 11. DataFrameCAN implementing

**Algorithm for sending CAN Data Frames By Publisher (CAN Dispatcher)**

```
1. Output :
     DataFrameCANListSortedByPriority
2. Sort of Topics By TopicQOS (Deadline)
       2.1 For each Topic :
       2.1.1 Sort of its DataWriters
3.  Generation  of  CAN  Data  Frames  list
(Sorted By Priority QOS)
 4.  Use  of  CAN  Dispatcher  to  send  Data
Frames to its subscribers.
```

**Algorithm for receiving CAN Data frames By Subsribers**

```
1.Input :
  DataFrameCANListSortedByPriority
  //Each Subsriber notify its Listeners By
Constructing an Event Object
2. For each DataReaderListeners :
       2.1. Read DataReaderQOS
       2.2. Search of the same QOS from
       DataFrameCANListSortedByPriority
       2.3      Call      of      DataReader
       on_Data_available method
```

Figure 12. Simulator algorithms Implementation

## 6. Conclusion

ture of real-time distributed systems, allowing simple and effective development of distributed applications. With the work described herein, the CAN bus has been rendered more usable in the field of distributed DCPS systems. We tried to design a DDS over CAN simulator which interacts with the main actors described by the DDS specification. This interaction aims to integrate the DDS QoS parameters to improve the consistent data delivery and to optimise network behaviour. This interaction aims to integrate the network resources control to high level middleware and thus enabling a new generation of flexible DRE applications that have more precise control over their end-to-end resources. The priority based mechanism and the EDF scheduling strategy used within the context of this work is adapted with soft real-time communication system.

One promising research direction is to use DDS not only for the DRE systems but also within the network on chip (NoC). However all QoS parameters described by the specification must be implemented before. We recall that we are limited in this work to QoS related to the real-time characteristics.

## 7. References


[1] G. Blair, G. Coulson, P. Robin, M. Papathomas, An Architecture for next generation middleware, Proc, 4th Annu, IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, London, England, 1998.

[2] OMG, "Data Distribution Service for Real-Time Systems Specification", March 2004.

[3] ROBERT BOSCH GmbH, CAN Specification version 2.0 (1s edition, 1991).

[4] First International Workshop on Data Distribution for Real-Time Systems, In conjunction with International Conference on Distributed Computing Systems, May 2003.

[5] M. Karakaya, O. Ulusoy, Evaluation of a Broadcast Scheduling Algorithm, Lecture Notes in Computer Science, Springer-Verlag, v. 2151, 2001.

[6] P. Xuan, S. Sen, O. Gonzalez, J. Fernandez, K. Ramamritham, Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments, Proceedings of the Fourth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'97), 1997.

[7] A. Bestavros, Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems, Proceedings of the 1996 International Conference on Data Engineering, New Orleans, LA, March 1996.

[8] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, D. Estrin, Data-Centric Storage in Sensornets, First Workshop on Hot Topics in Networks (HotNets-I) 2002.

[9] F. Ye, H. Luo, J. Cheng, S. Lu, L. Zhang, A Two-Tier Data Dissemination Model for Large-Scale Wireless Sensor Networks, MOBICOM'02, September 23-28, 2002, Atlanta, GA.

[10] Y. Yao, J. Gehrke, Query Processing for Sensor Networks, Proceedings of the 2003 Conference on Innovative Data Systems Research, Jan. 2003.

[11] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In Proceedings of the Second International Conference on Mobile Data Management, 2001.

[12] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, Energy-Efficient Communication Protocol for Wireless Microsensor Networks, In HICSS '00, January 2000.

[13] B. C. Lu, B. M. Blum, T. Abdelzaher, J. A. Stankovic, T. He, RAP: A Real-Time Communication Architecture for Large-Scale Wireless Networks, Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02), 2002.

[14] T. Abdelzaher, J. Stankovic, S. Son, B. Blum, T. He, A.Wood, C. Lu, A Communication Architecture and Programming Abstractions for Real-Time Embedded Sensor Networks, Proceedings of the First International Workshop on Data Distribution for Real-Time Systems, Providence, RI, May 2003.

[15] S. Kim, S. H. Son, J. A. Stankovic, S. Li, Y. Choi, SAFE: A Data Dissemination Protocol for Periodic Updates in Sensor Networks, Proceedings of the First International Workshop on Data Distribution for Real-Time Systems, Providence, RI, May 2003.

[16] S. Bhattacharya, H. Kim, S. Prabh, T. Abdelzaher, Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks, Proceedings of the First International Conference on Mobile Systems, Applications and Services, San Francisco, CA, May 2003.

[17] PrismTech, http://www.prismtech.com.

[18] H. V. Hag, OpenSlice Overview, white paper, 2006.

[19] S. Lankes, A. Jabs, and T. Bemmerl, Integration of a CAN-based connection-oriented communication model into Real-Time CORBA, Proc, IEEE International Parallel and Distributed Processing Symposium, Proc, 11th Annu, Workshop on Parallel and Distributed Real-Time Systems, Nice, France, April 2003.

[20] Real-Time Innovations, http://www.rti.com

[21] Thales Netherland, http://www.thales-nederland.nl/

[22] J. H. van 't Hag Data-Centric to the Max - The SPLICE Architecture Experience, Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), May 19 - 22, 2003.

[23] Salem Hasnaoui, "Implementing and Evaluating the Publish/Subscribe paradigm Using the OMG's DDS/DAIS Specifications. Application to IDEC's Micro3 PLC Computer Link System ", Journal of Control and Intelligent Systems ACTA press, Vol. 36, No. 1, 2008.

[24] T. Guesmi, S. Hasnaoui and H. Rezig, "Network Priority Mapping Using Dynamic RT-CORBA Scheduling Service", International Revue On Computers Software, September Issue, ISSN 1828-6003.

[25] T. Guesmi, S. Hasnaoui and H. Rezig, "Using RT-CORBA Scheduling Service and Prioritized Network Traffic to Achieve End-to-End Predictability", the 2006 International Conference on Communications In Computing (CIC'06), 26-29 June 2006, USA.