An IoT Middleware Framework for Industrial Applications

Nicoleta-Cristina Gaitan[1,2], Vasile Gheorghita Gaitan[1,2], Ioan Ungurean[1,2]
[1]Faculty of Electrical Engineering and Computer Science, [2]Integrated Centre for Research, Development and Innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for Fabrication and Control (MANSiD)
Stefan cel Mare University of Suceava, Romania

***Abstract*—Starting from the RFID and the wireless sensor networks, the Internet of connected things has attracted the attention of major IT companies and later, of the industrial environment that recognized the concept as one of their key axes for future growth and development. The implementation of IoT in the industrial environment raises some significant issues related to the diversity of fieldbuses, the large number of devices and their configuration. The requirements related to reliability, security and real-time are very important. This paper proposes an industrial IoT and communications at the edge framework which has some outstanding features related to: the easy integration of fieldbuses and devices used in industrial environments with automatic configuration features, integration of multiple middleware technologies (CORBA, OPC and DDS), the uncoupling of the industrial activity from the publishing data on the Internet, security at different levels of the framework. Another important feature of the proposed framework is that it is based on mature standards and on open source or public implementations of these standards. The framework is modular, allowing the easy integration of new fieldbus protocols, middleware technologies and new objects in the client application. This paper is focused mainly on CORBA and DDS approaches.**

***Keywords*—Internet of Things; Middleware; CORBA; ACE ORB (TAO); Data Distribution Service**

I. Introduction

KEVIN Ashton, from the MIT Auto-ID Center, was the first who proposed the term "Internet of Things" (IoT), referring to the connection between the information provided by radio frequency identifiers (RFID) and the Internet [1]. Quickly, the interest in the Internet of connected things caught the attention of governments and IT companies which have recognized the concept as one of the key axes for their future growth and development [2]. An increasingly accepted definition of the IoT was provided in [3]. In this definition, the emphasis is placed on virtual and physical “things” which: use intelligent interfaces; are fully integrated into an information network; have identifiers, physical attributes, and virtual personalities using a global infrastructure network with dynamic configuration (mobile), auto-configuration facilities, and interoperable communication protocols.

The potential growth of IoT technologies has led to increased interest in their use in various industries, where devices, machines, sensors, or simple things communicate with each other using standard Internet technologies [4]. It can be stated that the real value of the Industrial IoT (IIoT) is the availability of ubiquitous information and consequently, the decisions that can be made from it. An IIoT platform must validate the sharing of dispersed and ubiquitous data in an efficient and timely way for the web, cloud, desktop, embedded, and mobile applications. Therefore, IIoT can be defined [5] as the connection between the sensors from the physical world, devices and machines on the Internet and, by applying a thorough analysis using the software, the transformation of massive data into powerful insight, and intelligence.

It is becoming increasingly clear that the industry needs a functional and useful architecture for the Industrial Internet of Things (IIoT), which should include the recent progresses and novelty technologies in the field. Such an architecture should be easily understood and, at the same time, complete. Most projects and specialized literature are focused on how "things" can be converted so that they can be connected to the Internet through the addition of intelligence and connectivity, for instance by using the RFID technology for things/objects in everyday life [6][7]. Beside the RFID technology, they also take into account the wireless sensor networks. These architectures can be found in [8][9][10][11][12]. An important issue of this solution is security [13].

Although the issues listed above are essential for the IoT, it can be considered that in addition of RFID, wireless communication, sensors and actuators as IoT things, it can be added devices and machines with wired communication in order to define IIoT things. Furthermore, it can be pointed out that industrial automation involves difficult requirements regarding communication and the ensuring security and reliability. These requirements must be met by IIoT from the beginning. Currently, the implementation and operation of the complex production processes or of the Internet applications (Internet-enabled) requires time and a manual network setup that is susceptible to errors. This situation is generated by the need to ensure a high level of determinism, safety, and security during the production process and to avoid both critical security failures and costly production interruptions. These objectives should be IIOT-specific, including a high level of automation for the network configuration processes (including the fieldbuses pertaining to the industrial environment).

In this paper, it is proposed an IIoT framework organized on three levels, based on the three observations outlined above (italic): the device that integrates the hardware (sensors, actuators, RFID) in order to sense/control the physical world and to acquire data, middleware for data transport and an application which provides the means to interact with the user and other IoT applications [14]. The proposed framework can be the edge that bridges the information technologies and world of things, where the available resources in the cloud cannot be directly accessed [14]. In this case, the operational technologies are the fieldbuses with their features that represents additional challenges. At the low level, the framework understands different network topology, and data protocols that will be found into the world of things. This contains solutions for automatically discovery and identification of the real industrial things, data associated and to be able to perform storage at high-frequency updates. At the high level, the framework collects the data and sent it to the cloud via IIoT standards. In the CISCO visions [15], the framework represents the Edge Computing (that is also called Fog Computing).

The framework is in accordance with the IIoT definition which was presented previously. The solution uses OPC (Open Platform Communication) [16], OPC .NET [16], OPC UA [16], TAO [17] and DDS (Data Distribution Service) standard [18][19] are used as middleware (an important component in IIoT) in order to show the data at enterprise level and DDS for the external enterprise interoperability. This article mainly takes into account the implementations based on TAO and DDS. In the process of defining the framework, three great challenges arise: (i) the large number of fieldbuses, description of devices and automatic configuration; (ii) middleware choice and provision of real-time services; and (iii) separation of the industrial activity from the operations, for the sake of data publication and subscription on the Internet, and incorporation of different types of technology. The proposed framework can be used in smart factory but the utilisation can be extended for smart home, smart buildings, smart living, and smart city.

Furthermore, this paper is organized as follows: Section II briefly presents different architectures proposed for the use of IoT in industry. Section III presents our proposal for an IoT based on TAO for the industry field. Section IV presents the test performed in order to compare the bandwidth used by a TAO-based server with one based on OPC DA, OPC UA and OPC.NET in a local network. Section V presents a comparison between TAO/OpenDDS and OPC UA as support for IIoT. The final conclusions are drawn in Section VI.

II. Related Work of the Industrial IoT Architecture

When a new IIoT architecture and a practical implementation are proposed, a natural question which arises is: what are the existing solutions? The literature specialised in the field is very poor in such solutions because IIoT is at the beginning. A courageous attempt is made in [20]. The authors, relying on a rich bibliography, tried to understand the current status and the future research opportunities related to the use of the IoT concept in industry. Only Section V strictly refers to the applications of IoT in industry, fields such as healthcare service, food supply chain, transport and logistics, and firefighting, which are more in the field of services and infrastructure and not industry, are being taken into account. The only industrial sector already addressed is mining production [21][22]. Our bibliographic studies have led to similar conclusions. There are few articles related to IIoT and those are strictly focused on specific applications. In what follows, we will briefly present some concerns present at institutional level or which are covered by research projects.

In Germany, the IoT is associated with the field of production and logistics through the term "Industry 4.0"[23], and the grounds are being prepared for a new social and technological revolution which will drastically change the whole industrial environment. Industry 4.0 is a sophisticated change of the entire chain of values: communication, planning, logistics, and production. Due to the success it recorded in the fields of information and communication technologies (ICT) (currently 90% of all manufacturing processes are already supported by ICT) and embedded systems, (strong autonomous microcomputers) either connected to each other or to the Internet, wired or wireless, it will lead to a convergence between the physical and the virtual (cyberspace) world. This convergence takes the form of a Cyber- Physical Systems (CPS), term used international to describe Industry 4.0 concept. With the development of IPv6 standards, there are now enough addresses to allow, for the first time, the networking of resources, information, objects, and people, in order to create the Internet of Things and Services. The proposed architecture is set on four levels (from bottom to top): Internet of Things, Internet-based System & Service Platforms, Internet of Services and Applications. More details can be found in [23].

Another interesting research project in the IIoT field is the IoT@Work [24]. The project focuses on the exploitation of IoT technologies in the industrial and automation sectors. The architecture proposed in this project has five horizontal levels and three vertical planes. The horizontal levels refer to: Field/Control Infrastructure & Network, Device and Network Embedded Services (auto-configuration, device semantic, network management), Device Resource Creation & Management Services (abstraction, context/dependencies), Application Level Middleware Services (commissioning, composition, adaptation), and Automation Applications. The vertical planes are the following: communication plane, security plane and management plane. The project proposes the following technologies for the IIoT: Directory Service, Auto-Configuration of Real-Time Ethernet, Event Dispatching (Event Notification Service), Capability-based Access Control, Complex Event Processing, Network Slices, and Embedded Access Control. More details on the proposed architecture and technologies can be found in [24].

An interesting discussion is launched by Herman Storey (co -chair ISA 100), Rick Bullota and Daniel Drolet in [25]. The discussion begins with the observation that IIoT should primarily provide security, robustness, and punctuality as far as the requirements of automation networks are concerned and, secondly, remote access. The IIOT proposed architecture has four horizontal levels: multiple physical media and link layer, IPv6/6LoWPAN common network layer, more communication stack layers and multiple applications layer. Vertically, the architecture has two levels: Common time and Common network management and security. As an essential element, the IIoT must provide a way to integrate multiple physical environments and multiple applications in a single industrial network system using common technologies. To integrate such a variety of communication and application environments, the IIoT must use IPv6 as a network protocol. IPv6 has an extension called 6LoWPAN which allows it, as a network layer, to be used for low power networks or limited bandwidth. Although it was designed for battery-powered wireless devices, it may be used for wired networks as well. ISA 100.15 published a document which provides models and concepts for architectures adequate for IIoT.

Following the analysis of the three architectures presented above, it can be said that currently, there is a low degree of standardization. Efforts are being made to achieve an IIoT standard (Industry 4.0, ISA 100). The IIoT is a different IoT from the non-industrial ones due to the special characteristics of the production processes. Except for Industry 4.0, IIoT architecture is based on ground level devices, which are interconnected via fieldbuses and which have access points to local networks and the Internet, while on the upper level it has specific applications. Intermediate levels ensure services for the safe transport of information. In addition to the horizontal levels, there may also be vertical planes, able to ensure management and security, time management, and so on. The expectations of IIoT refer to the possibility that devices, machines, and other objects could interact with each other without relying on human intervention to achieve added value. Among the most important requirements for IIoT [26], we can mention: reliability, robustness, reasonable cost, security and safety, easy use, low/no maintenance, optimal and adaptive set of features, standardization, integration capabilities, reach sensing and data capabilities, industry degree support, and services. The challenges faced by IIoT refer to IoT devices, lifetime and energy, data and information, humans and business.

III. The IoT Framework Proposed for Industry

In this section, it is presented the new proposed IoT framework for industry where devices, machines, sensors, or simple things must communicate with each other. This IIoT framework is composed of three levels (device, middleware, and application). The first level is the device level. It is composed of three elements, namely: the device which acquires data directly from the environment and can transfer this information using a wired or wireless network/fieldbus, the gateway which adapts the specific network protocol to the specific computer protocol used by the middleware in order to connect to the IIoT environments (which can also add real time facilities) and the software driver for the gateway device which adapts the information sent or received to/from the gateway in order for it to be compatible with the middleware. The middleware level is designed to provide data transportation inside the IIoT and it is based on the OPC, CORBA (with TAO implementation (The ACE ORB)) and DDS. The application level provides support for the implementation of the basic applications pertaining to the proposed framework and the level’s middleware objects which can be embedded in other IIoT applications [27]. The specific interoperability model is provided by the OPC and TAO, while the global interoperability is ensured by the DDS middleware standard [28].

A. *The motivation of the proposed framework*

In order to motivate the proposed framework, we can begin from the question: is it a new technology? The answer is that it is a new vision related to the reorganisation of a sum of existing technologies in order to satisfy new requirements concerning the future development of the industry.

Regarding the device level, the following major problems were considered: there are different physical and data link layers which respond to different requirements of specific applications in the industry field; at the extremity of the global network, there are fieldbuses that are intended to acquire information from sensors and transducers, and to emit commands via actuators; and that all these fieldbuses must have common support for IPv4/IPv6. For this level, a gateway device is defined, one which must implement the gateway function [29] in order to transfer the information to the higher level. It must transform the process-specific information into information useful for the higher level [30] and it must provide real-time behaviour at fieldbus level. Furthermore, a description method for devices, recognised by all partners who require information about devices, must be developed. Network/fieldbus configuration for acquisition of information from the process is a time-consuming and expensive operation which means that tools capable of automating this operation must be created. In the fieldbuses area field, there is currently a multitude of standards (and perhaps new standards will appear in the future) which means that, consequently, the framework must support the integration of new protocols.

The middleware level has the important task of transporting information between different nodes placed in the Intranet, Extranet, and the Internet. This level implies important design decisions. Standard-based middleware’s were taken into account due to their stability and impact on the industry. Since the OPC specifications are specially designed for industrial applications, a first major question is: why TAO and DDS? A second question may be: why not just DDS? The short answer to the first question is: the OPC specifications have no explicit real-time requirements and use the client-server paradigm, which is less suitable for data centre frameworks of the publisher/subscriber type; and answer to the second question is: TAO is better prepared for real time. Further, these two answers are expanded.

A very interesting discussion on the utilization of standards for real-time distribution middleware is presented in [31]. The authors, out of several distribution models, chose those which are based on the standard, are mature, stable and with impact on the industry; namely: CORBA/RT-CORBA, Distributed System Ada Annex (DSA), Data Distribution Service for Real-time System (DDS) and Distributed Real-Time Specification for Java (DRTSJ). Even though the authors of [31] do not provide a verdict or have not carried out a ranking, however, a classification can be made.

CORBA/RT-CORBA has the following advantages: it is based on a very mature technology, one involved in a wide range of applications [31], such as Software Defined Radios [32] and Industrial Robotics [33]; RT-CORBA entities validate the development of critical real-time applications; from the point of view of scheduling, the RT-CORBA provides static scheduling based on Fixed Priority Scheduling (FPS), the use of threads as schedulable entities, control of the competition degree on the servers using thread pools, deterministic access to shared resources, the use of different scheduling policies, and the use of distributable threads as a schedulable entity; as far as network resource management is concerned, it provides mechanism for the fine-tuning of network properties, it uses private connections and definitions of priority-banded connections; it is the only standard which provides mechanisms for the specification of scheduling parameters which may be used during execution; facilitate interoperability between implementations (GIOP - General Inter-ORB Protocol); TAO implementation is the most popular and updated open-source implementation for RT-CORBA. As disadvantages of RT-CORBA, we can mention: unlike the CORBA specification updated in [34], RT-CORBA is not currently in the attention of the Object Management Group (OMG), the last update being performed in 2005 [35][36]; it does not take into consideration the network scheduling; it uses TCP/IP stack which means that even the use of Ethernet switches is unsuitable for implementation of hard real-time systems; TAO implementation does not provide synchronous protocols (it is based on the operating system); it does not implement the priority transforms model, the use of buffers to store remote requests in thread pools nor the borrowing of threads among thread pool lanes.

The DDS has the following advantages: it is considered a mature technology involved in several real-time applications [31] in the fields such as Defence [17], Automation [37], and Space [38]; supports anonymous and asynchronous dissemination of information; has specific requirements for distributed applications such as control systems, sensor networks, and industrial automation systems; it is a data-centric middleware [18] and, therefore, it is aware of the contents of the interchanged data which can be directly managed; it provides multiplatform and multi-language support; the types of shared data can be defined by using IDL language [34]; interoperability between different implementations is provided by DDS Interoperability Wire Protocol (DDSI) [39]; it is a recently updated specification, OMG provides specification for the Extensible and Dynamic Topic Types [40], which provides support in order to define and modify dynamic (on runtime) data for the extension and evolution of systems based on DDS; the DDS model defines a strongly typed Global Data Space where publishers (Data Writer (DW)) can write (provide) data and subscribers (Data Reader (DR)) can read (consume) data allowing the middleware to focus on obtaining data independent of their origin; the standard was explicitly designed for distributed real-time systems; specifications define a set of QoS parameters in order to configure non-functional properties for each entity and allow the change of some of them during an operation; a subset of QoS parameters allows the control of temporal behaviour and improves the application predictability; it defines different mechanisms meant to validate the communication between entities (polling, synchronous mode and asynchronous mode for the DR entity) and provides the opportunity to notify the application by Polling, Listeners, Conditions, and Wait-sets; there are both commercial (CoreDX or RTI-DDS) and open source (OpenSplice or OpenDDS) implementations. Among the DDS disadvantages, we can mention: there are no evaluations in detail done on the DDS real-time performance (an attempt can be found in [41]); it does not explicitly addressed the scheduling of threads at processor level; it is oriented on IP networks and not on the real-time networks (still lists a set of requirements for network support); considers only network policies based on fixed priority scheduling and excludes any other type of predictable network used in industry; some internal middleware operations generate meta-traffic thus introducing an override that must be taken into account in the analysis of behaviour in time; DDSI has an indefinite number of sub-messages; there is still no profile for safety-critical applications.

The DSA and DRTSJ are not competitive for real time as CORBA/RT-CORBA and DDS. The DSA [31] was specifically designed to support predictable applications and several features, which ensure determinism, are left to application implementation; while the DRTSJ [31] specification is not complete, there are still problems which were not addressed and there is no formal DRTSJ specification (only a draft). On the other hand, all these protocols and their implementations for real-time communication use IP-based networks. Even if local networks that use switches are used, real time is not easily achieved.

For the application level, the design issues taken into account are: easy embedding and integration of several technologies (OPC DA, OPC .NET, OPC UA, and CORBA); default communication between application objects by defining a "software bus" so that the application objects communicate with each other and the implementation, at the current level, of the gateway function between different technologies; decoupling of the company’s activities and specific production processes, which requires a high degree of security; the publication of some information on the Internet; platform-independent communication between the instances of several applications; establishment of a connection with the usual databases which benefit from a specialized middleware for data communication.

The IoT framework of the system proposed in this article, in order to integrate IoT in the monitoring and control of the industrial processes, is presented in Fig. 1. The proposed framework is based on the OPC specification, DDS and CORBA middleware (TAO implementation). Furthermore, the framework will be presented from the point of view of CORBA and DDS middlewares. These middlewares were used because they allow the development of applications distributable on the Internet. In industry, CORBA middleware is not widely used although there is the DAIS [42] standard which describes how to develop SCADA applications based on CORBA. In the proposed framework, new TAO servers and clients are considered supplementary uses, which, just as DAIS, are based on the OPC DA 2.05 specification. Our solution is easier to implement compared to DAIS.

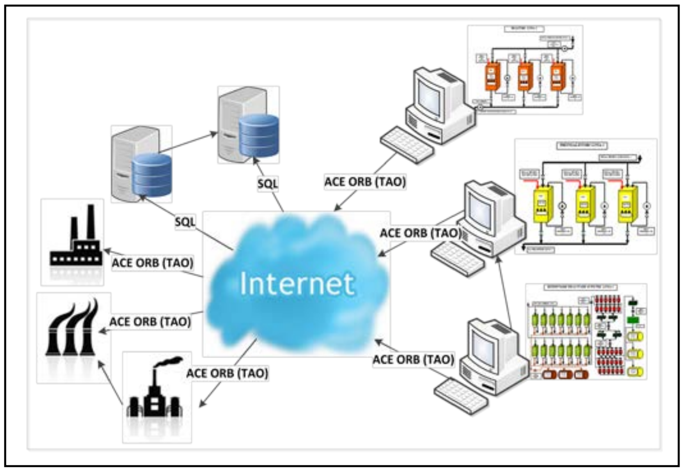

Fig. 1. Distribution on Internet of the proposed framework

From the point of view of implementation, the proposed framework consists of two main functional modules: the data-acquisition module (which will be referred to as the server module) and the Human-Machine Interface (HMI) module (for the information retrieved from the server modules) which will be referred to as the Human Machine Interface - Process Control and Monitoring (HMI-PCM) module, and it is mainly a client application for TAO and OPC servers. DDS is implemented as an object in the HMI-PCM. By using the TAO, the information acquired from the industrial process can be distributed on the Internet, in a client-server manner, as noticeable in Fig. 1. A functional (complex) system can be composed of multiple servers and multiple HMI-PCM clients. A HMI-PCM client can connect to multiple server modules and database servers, as described in the following sections. Clients can generate history based on the data read from the server, history stored in a database which can be consulted later by the client who generated this history or by other clients.

B. Server module

The architecture of the server module is shown in Fig. 2. This architecture is structured on three main levels. On the lower level, we have the drivers which acquire data from the fieldbuses and store it on the cache located on the upper level. This level is integrated in the device level of the IIoT framework. Its main role is the implementation of the acquisition cycle which is specific to the fieldbuses protocol used for communication. On this level there are more software modules, each module specific for one fieldbus. Furthermore, these modules receive data from the top level, which will be sent to the fieldbuses (e.g. commands for actuators). These modules receive all the data which must be updated continuously from the top level (data that is in at least one client's subscription list). This data is included in the acquisition cycle implemented in the drivers. Furthermore, these modules implement mechanisms for the data read on request (asynchronously read). They rely on a running platform (Linux or Windows) and are developed as independent modules (as libraries). This allows the development of new drivers without recompiling the other server modules. Between this level and the upper level, there is a well-defined interface that allows the integration of drivers for new fieldbus protocols (API 1 from Fig. 2).

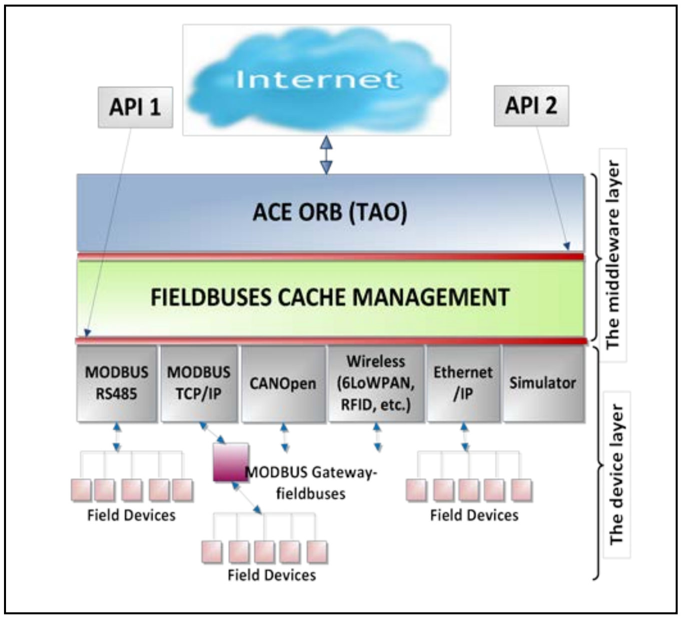

Fig. 2. The server module architecture

On the intermediate level, we have the Fieldbuses Cache Management (FCM) module which deals with the management of the cache which stores the data read from the fieldbuses, and which is also developed as an independent module (as a library). This memory cache is necessary to achieve a rapid response to the requests received from the upper level. The cache memory is a resource shared by several threads and has all the access control mechanisms implemented to ensure data consistency. Furthermore, this module stores a list of data on which clients are subscribed to ensure continuous updating of the cache (data update is provided only for that list of data). The data received from clients, which must be submitted to the devices connected on fieldbuses, are stored in the cache and are forwarded to the appropriate network driver. This level is part of the middleware level. Between this level and the upper level (the server itself), there is a well-defined interface which allows the adaptation of the FCM to any desired type of server, including TAO server (API 2 from Fig. 2).

On the top level, we have the server which provides support to access the cache with both read and in writing operations, in other words, the access to field devices connected to networks. Furthermore, the server integrates the TAO middleware which provides services for the transmission/reception of data to/from the HMI-PCM clients. To ensure these services, a CORBA IDL interface was defined, one which has been integrated into the server and the client modules. The interface is based on the OPC DA 2.05 classical specification. So, four interfaces were defined, namely: DataServer, an interface with a Register (server connection) and DeRegister (disconnected from server) methods; IServer interface with Addgroup, RemoveGroup, and SetState methods (edits the properties of the group); IGroup interface with AddItems and RemoveItems methods; IUpdate interface with OnDataChange (updates data to the client group) and Disconnect (server being offline) methods; IBrowse interface with BrowseAddressSpace (accesses the server address space), ChangeBrowsePosition (browses the address space server), GetItemID (takes over the address space identifier of server), QueryAvailableProperties (reads the properties of an Item), SyncRead (synchronously reads the value and quality of a list of items, from the cache or device), and SyncWrite methods (synchronously writes the value and quality of a list of item, from the cache or device). We detail the implementation based on TAO because it is a less used a solution in industrial environments compared with servers based on OPC specifications.

C. HMI-PCM – Human Machine Interface -Process Control and Monitoring

The client application (HMI-PCM) is an environment that can instantiate many objects (controls). There are three types of objects: graphical objects, middleware objects and expression objects. They expose data members in the HMI-PCM environment. The data members can be interconnected in order to transfer data between objects, or can be used in different math expressions to which other objects can connect (subscribe) by using a standard interface (API 3 from Fig. 3). Middleware objects connect to data providers (servers) based on different middleware packages (OPC.NET objects to transport data from/to OPC.NET data servers, OPC DA objects to transport data from/to OPC DA servers; OPC UA objects to transport data from/to OPC UA servers; TAO objects to transport data from/to CORBA servers). The architecture of the HMI-PCM module is presented in Fig. 3.

OpenDDS is an open source implementation of the DDS specification based on TAO. The DDS objects from HMI-PCM environment ensure the interoperability between different HMI-PCM applications running anywhere (on the same computer, the computers interconnected throughout local network or computers interconnected throughout the Internet). The objects can expose the HMI-PCM address space, including middleware objects that partially or fully expose the server address space (see subsection D).

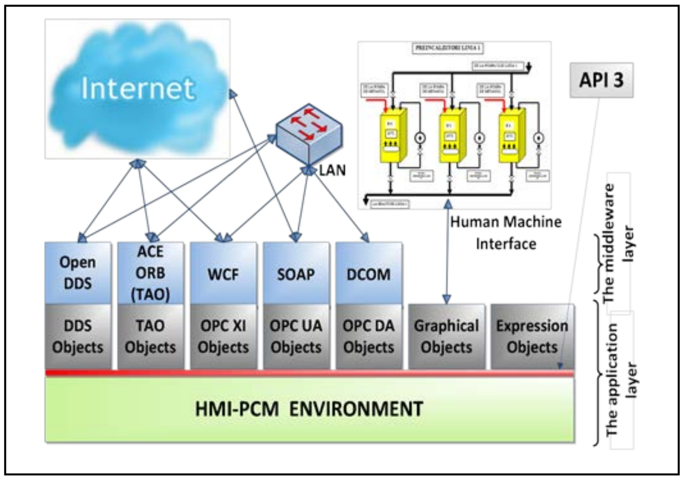

Fig. 3. The HMI-PCM module architecture

The most important feature of this application is that it allows the interconnection of objects in the HMI-PCM. Each object has data members that can be connected to each other or to the data members of other objects from the HMI-PCM. Thus, to display the data from the server, a graphical object is used, one that connects to the TAO objects that are connected to these servers. With this feature, the HMI-PCM application can be easily configured according to the user's requirements and preferences. Another important feature of the HMI-PCM is that new objects can be added as dynamic libraries. They must comply with HMI-PCM standard interface (API 3 from Fig. 3) that enables communication between the HMI-PCM objects (objects derived from a basic object). So, it is not necessary to compile the whole clients (only the object added).

D. Implementation considerations

The server is developed and implemented as an application in C++. For each fieldbus, there is a library which implements the function specific to the fieldbus. It was implemented a library for MODBUS RTU (with a RS485–RS232 interface), a library for MODBUS TCP/IP and a library for CANOpen (with a USB-CAN interface). The libraries for EtherCAT and Ethernet/IP are under development process. Since there are many Modbus TCP/IP gateways to other fieldbuses, these systems can be easily integrated into the proposed framework (should be considered the differences in terms of real time between fieldbuses and MODBUS TCP / IP because TCP/IP stack is best effort type and not real time). For the transport protocol between server and client, the following protocol was employed: IIOP (default) Internet Inter-ORB Protocol, SHMIOP - shared memory transport protocol, IIOP over Secure Sockets Layer (SSL), HTTP Tunnelling Inter-ORB Protocol, and ZIOP – IIOP with compression).

Due to the modular software architecture of the server (see Fig. 2), servers based on more middleware types were developed, while this paper deals with the server based on the TAO middleware (version 6.2.5). The server will expose data as a collection of industrial networks, each network having a collection of devices. Every device connected to an industrial process can be seen as a collection of objects. For this reason, a dictionary of objects was developed, managed by the FCM, exposing all the capabilities of the devices. Each object can have multiple data members and each data member can be characterized by properties such as value, data type, access rights, or other property that can be defined by the user based on the application. The content of the object dictionary (data provider) forms the address space of the server. Each middleware object will expose this address space to the client. A natural question is how to create this address space? FCM has on the upper level a defined standard interface for server connection (API 2 from Fig. 2), and another one at the bottom for connection to the fieldbus-specific drivers (API 1 from Fig. 2). Any driver that implements this interface is loaded without recompiling the entire application.

At this point, another question appears: how are the system devices described? Among the various solutions (EDDL - Electronic Device Description Language, FDT –Field Device Tool, FDI-Field Device Integration, EDS - Electronic Data Sheet), for simplicity reasons, a solution was adopted, based on the CiA DS 306 D3 v1.3 specification (EDS). This specification has been extended to support Modbus, M-Bus and ASCII-DCON protocols in addition to CANOpen. Modbus TCP/IP gateway connects to other protocol implementing devices, such as Profibus, Profinet, EtherCAT, EP PowerLink, Ethernet / IP, LonWork, etc. From a device, one cannot get more than the information that is defined in the corresponding EDS. For example, for the Modbus protocol, a new section called [Communication] was added. This section of the EDS file describes the commands required to access objects like:

[IndexObject]: Request: FC: SFC-x: ADR: L-x:E: ADR:L-x:E: ADR, L-x:/

Response: FC: SFC-x: ADR: L-x:/

Where: IndexObject – Process Data Object (PDO) or Service Data Object (SDO) that describes the data. Request: the format of request commands: FC – function code; SFC – sub-function code; ADR – address; L-x – length or count, x = number of bytes of this field; E – The extension of the commands. Response: the response format that is optional. For the functions of the MODBUS protocol, the answer can be built depending on request commands. If the PDO or SDO objects have a defined separate area for read and write operation, subsections [read] and [write] may be used. “:” - fields’ separator (if a field is missing from a MODBUS command/ response, only a separator, “/” – terminator is used).

In the (automatic or manual) configuration process, a file is created in order to attach a driver to each fieldbus (a specific dynamic library) and an ID and an EDS file to each device from the fieldbus. This file is used by the FCM, which sends the path to the EDS files of the active devices from the fieldbus to the driver. There may be several fieldbuses of the same type and more identical devices in one fieldbus. A configuration file associated with the server and built on EDS files contains the entire tree structure of the information that can be accessed and forms the address space of the server. This address space can be accessed by the TAO object from the HMI-PCM application through the IDL interfaces defined at the end of subsection III.B.

Once the server address space is defined, the server will expose this information to the clients, using the interfaces defined in IDL (see the end of subsection III.B). The main implementing objectives of the server refer mainly to the service name, client management, client-associated group management, group-associated items management, updating groups, reading and writing items, browsing in the address space, security information, and QoS.

The HMI-PCM is developed and implemented as an application in C#. Each object (see Fig. 3) is a library which exports a class derived from a base object. For the TAO object, a wrapper was used to marshall data from C++ to C# (TAO is developed for C++ application). The HMI- PCM application is developed in C#, as it offers the possibility of rapidly developing graphical applications and for productivity reasons. The HMI-PCM application is very interesting, allowing the communication between servers implemented with different technologies. Each server has one or more simulation drivers (a client can write or read to simulating some functionalities which can read or write by other clients). In addition to their role of simulation, these drivers allow the implementation of a relay function (gateway) between different types of servers. For example, suppose that the HMI-PCM has activated two middleware objects, one for OPC UA (data profile) and one of TAO type. A TAO user wishes to expose, to TAO clients, the nodes of the OPC UA. Firstly, it must create an EDS file for the simulation driver for the TAO server with the desired objects that are visible from the OPC UA object (the compatibility of data types must be ensured). The objects exposed by the TAO object based on the EDS file will be found in the FCM dictionary of objects. In the HMI-PCM client, any item of the OPC UA object (from the ones chosen and described in the EDS file from TAO) can be connected to a corresponding item exposed by the TAO server based on the EDS file for the simulator (read or write - IN or OUT).

All the TAO clients can read or write properly from/in the items exposed by the simulator. There can be any number of simulators (depending on the host system resources). This type of relay can be attained between any of the middleware objects using a simulation driver and its attached EDS file. Connection can also be made directly, with the specification that an item should be output (or bidirectional) and the other input (or bidirectional), and the data types must be compatible. In addition, one can connect an intermediate expression object which can operate on source value using a mathematical expression.

For low power communication stack, there is the MICRO PROFILE and COMPACT PROFILE as part of CORBA/e (and it is implemented in TAO), while reliable communications and Internet-enabled communications are provided by TAO through transport protocols and naming service.

E. Security

Security features are presented at different levels of the proposed framework. In general, at the fieldbus protocols, security features are not provided, because they introduce an additional overhead and are non-deterministic components. In order to use the FCM component, the server must authenticate throughout a unique identification key. In the absence of authentication, the exported functions of the FCM module do not work correctly. The same thing happens with the fieldbus drivers. The current security level of the application is sent to the FCM in order to enable/disable the controls from the windows of the network manager, the connection manager, and from other configuration windows exposed by FCM and fieldbus drivers. The server application has an access panel that requires a user name and a password in order to view and change configuration parameters of the fieldbuses. Users are divided into groups, for users, manager, administrators and guests, each group having restricted access to the functionalities of the server, except for the administrator group. The server configuration is stored in an encrypted XML file (hidden somewhere in the system). The same vision is applied to the HMI-PCM application.

At the middleware level, in TAO there is the possibility to comprise messages (using pluggable ZIOP protocols) and to secure the communication (using SSLIOP pluggable protocol that is based on SSL). In the original DDS specification, related to the security, only the following is specified: “the application could attach security credentials via the USER_DATA policy that can be used by the remote application to authenticate the source”. The new DDS security specification [43] (request for proposal) proposes interesting solutions based on Domains Secure and Confidential Topics. RTI has a wide range of security solutions such as: domain separation, access control and secure bridging; deep packet inspection; data filtering; secure operating system; secure transport; improved paradigm for secure distributed infrastructure [44]. OpenSplice ensure DDS Secure Networking Service and Access Control [45]. For OpenDDS, we integrated the SSLIOP (from TAO) through Extensible Transport Framework, in order to enable confidentiality and authentication.

OPC DA security for the communication is based on DCOM security, OPC. NET has different binding modes and types of authentication security modes depending on the type of binding (Named piped, TCP, HTTP Basic and HTTP WS) more types of authentication are being offered. OPC UA contains the philosophy related to the security in the specification, namely OPC UA part 2 - Security Model [46]. OPC UA is Secure-by-default, encryption enabled, and uses advanced certificate handling.

IV. Experimental Results

This section presents the tests performed for the proposed solution based on TAO (with 3 transport protocols: IIOP, SSLIOP, ZIOP) when it is used in a local network. First, the bandwidth used by the server based on TAO was compared with the one used by the server based on OPC DA, OPC UA and OPC.NET. Tests were performed in a network composed of eight computers, identical in terms of hardware and software, and a switch with 100Mbps Ethernet ports. Each computer had an AMD Athlon (tm) 64 X2 Dual Core Processor 4200+ 2.21GHz, 1GB of RAM and a Windows operating system. On one computer (which will be referred to as the server), are executed in turn the data server based on OPC DA, OPC UA, OPC.NET, and the server based on TAO. All these servers use the same data provider (a simulator that generates random values for items and stores them in the cache memory of the server). For the experimental test, we used version 6.2.5 for TAO and the IIOP, SSLIOP, and ZIOP protocols. On the other computers, the HMI-PCM application is executed in turn with TAO, OPC.NET HTTP, OPC.NET TCP, OPC UA BIN (data profile), OPC.DA objects connected to TAO, OPC.NET HTTP, OPC.NET TCP, OPC UA BIN (data profile), and respectively, OPC DA servers. For the TAO objects, the IIOP, SSLIOP and ZIOP were used, as transport protocols. Clients will make a group/subscription/list (the names are specific to the used middleware) that contains 16 items/nodes whose data type is BYTE.

With Colasoft Capsa software package, the traffic speed on the server computer was measured. It should be noted that there is no network traffic generated by other applications (the LAN is not connected to the Internet). The software architecture of the tests performed is shown in Fig. 4.

The first test consisted in determining the transfer rate when data is updated at a rate of 100ms. The test results are shown in Fig. 5. In this figure, we can see that the bandwidth occupied when using TAO with IIOP and SSLIOP is higher than when using the OPC DA, OPC UA BIN and OPC.NET TCP, and smaller than when using the OPC.NET HTTP, but is lowest when ZIOP is used as transport protocol.

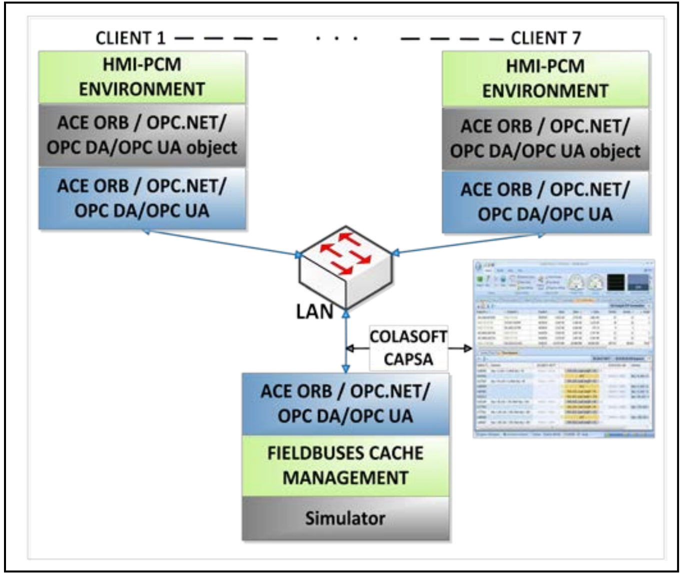
CLIENT 1
. . .
CLIENT 7
HMI-PCM ENVIRONMENT
ACE ORB / OPC.NET/ OPC DA/OPC UA object
ACE ORB / OPC.NET/ OPC DA/OPC UA
HMI-PCM ENVIRONMENT
ACE ORB / OPC.NET/ OPC DA/OPC UA object
ACE ORB / OPC.NET/ OPC DA/OPC UA
LAN
COLASOFT CAPSA
ACE ORB / OPC.NET/ OPC DA/OPC UA
FIELDBUSES CACHE MANAGEMENT
Simulator

Fig. 4. The software architecture of the tests performed

The second test consisted in determining the transfer rate when data is updated at a rate of 500ms. The test results are shown in Fig. 5. From this figure, we can see that the occupied bandwidth when TAO is used is higher than when OPC DA and OPC.NET TCP are used, and lower than when OPC.NET HTTP or OPC UA BIN is used. Unlike the previous test, the bandwidth occupied by TAO is much closer to the bandwidth occupied by OPC.NET TCP and OPC DA.

The third test consisted in determining the transfer rate when data is updated at a rate of 1000ms. The test results are shown in Fig. 5. As in the previous tests, the same approximation trend of the band occupied by TAO with the band occupied by OPC DA and OPC.NET TCP can be noticed.

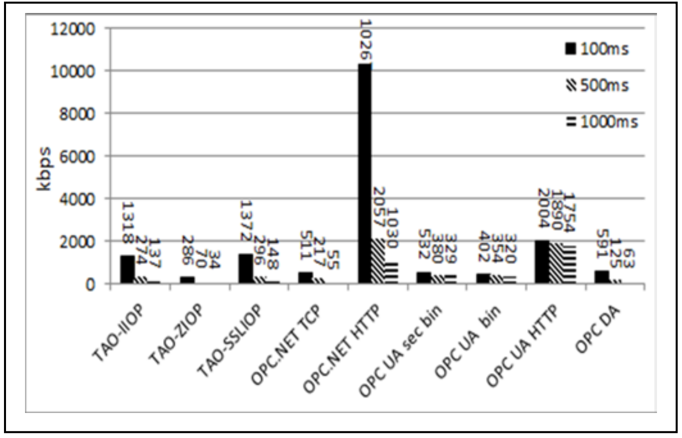

Fig. 5. Bandwidth occupied for a refresh rate of 100ms

Fig. 6 presents a synthesis of the 3 cases presented so far. An approximation trend of the bandwidth occupied by TAO with the bandwidth occupied by OPC.DA and OPC .NET TCP can be easily noticed. It should be noted that the tests were done in a local network, a framework widely used in the operation of industrial SCADA applications.

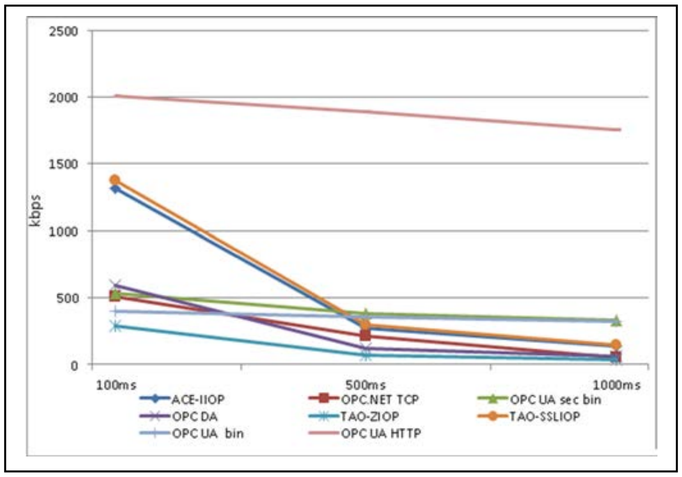

Fig. 6. Comparison for the bandwidth occupied

The proposed framework is designed to provide access to data via the Internet, where the response time cannot be guaranteed. It is unlikely to apply a refresh rate of 100ms for a client to connect to a server through the Internet, and very likely to use refresh rates of around 1000ms (in the Internet, this refresh rate cannot be guaranteed because the communication protocols are best-effort type depending on the network load).

The performances of the application based on TAO IIOP are very close to the performances of the applications based on OPC DA and OPC.NET TCP at an update rate of around 1000ms, but OPC DA is based on DCOM technology that works in a LAN network and OPC.NET TCP is dependent on .NET platform, based on Windows Communication Foundation. Furthermore, to use OPC.NET and to get the source code, you must be a member of the OPC Foundation. One advantage of using the TAO middleware is that it is an open source.

TAO with ZIOP transport protocol is the best because the messages are archived, but it does not provide any security mechanism. The use of security and encryption of the messages with SSLIOP transport protocol (based on SSL) introduces an additional overhead of the messages related to the IIOP transport protocol, which can be seen in the graphs, due to certificate exchanges and the increasing of the message size. The same difference can be seen for OPC.UA binary and OPC.UA binary with security and encryption of the messages. The use of the HTTP protocol leads to a significant increase of the messages that can be seen for OPC.NET HTTP and OPC.UA HTTP. In the case of the OPC.UA middleware, an important traffic generated by the keep alive mechanism (observed with Wireshark tool) can be observed. This traffic is much lower in the TAO implementation. From Wireshark tool, it can be seen that if the encryption or archiving mechanisms are not used, the data can easily be identified in the messages.

Table 1 presents the number of bytes of Ethernet frames and TAO for the three transport protocols (IIOP, ZIOP, and SSLIOP) sent by the server in order to update a group consisting of 1, 2, 4, 8, and 16 items. This information was obtained with Wireshark tool. As expected, the smallest frames are obtained by activating the ZIOP transport protocol. If the messages are small, and the size of the archived message (plus archived message header) is higher than the size of the original message (with IIOP), then the message is no longer archived and it is sent using the IIOP transport protocol.

TABLE I. THE MESSAGE FOR TAO TRANSPORT PROTOCOLS

	TAO-IIOP	**TAO-ZIOP**	**TAO-SSLIOP**
1 items	296B/1 frame	314B/1 frame	351B/1 frame
2 items	402B/1 frame	324B/1 frame	475B/1 frame
4 items	674B/1 frame	335B/1 frame	810B/2 frames
8 items	1174B/1 frame	357B/1 frame	1310/2 frames
16 items	2236/2 frames	397B/1 frame	2401/3 frames

From the point of view of the memory footprint, the working set for the server with TAO is about 11MB for IIOP, increased to about 36MB for ZIOP and reaches about 38MB for SSLIOP, while the processor load depends on the refresh rate of the items, reaching 8% for a refresh rate of 100ms. On the other hand, OPC UA has a working set that varies from 48MB (without encryption and security) and reaches about 174MB with encryption and security. The processor load at a refresh rate of 100ms is about 50%. This may be due to the development mode of the server, which is developed in C# and the code is interpreted, while TAO is implemented in C++.

V. CONCLUSION

In the rather poor landscape of IIoT architectures, the proposed framework can be a starting point, especially since efforts are being made to implement and to perform a practical demonstration of the proposed functionalities.

This model was referred as framework and not as architecture because it is concerned with the IIoT device platform that transport the specific messages (little data) and which, through the DDS objects, can connect to the IoT services and applications (big data).

The framework enjoys several powerful points. First, it is based on mature and very mature standards and it can say that it is highly standardised. At device level, a unified method to describe the devices based on the EDS specification from CiA was defined. It was extended, among others, for the MODBUS protocol. Currently, there are many MODBUS TCP/IP - other protocol gateways which have their own mechanism of describing the devices; it can be depicted by the EDS modified for MODBUS. This solution resolved the challenge related on the large number of fieldbuses. The standardised interface from the lower level of FCM is scalable, allowing the integration of drivers specific for other fieldbus protocols without recompiling the FCM module and the server. The presence of the objects' dictionary, which creates the server address space, is the reason for the decoupling (virtualisation) between server and the complexity of fieldbuses, and a unified way of describing them. The configuration interfaces of the fieldbuses have a semi-automatic behaviour (drivers identify the field devices and the display on the server objects which can be exposed, and the server automatically restores the last saved configuration).

At the middleware level, several technologies were selected and implemented (OPC, TAO, and DDS) which enable a proper adaptation to the specific application. The PCM-HMI application allows easy exchange of information between servers and, by implementing the DDS objects; it allows the publisher / subscriber a type of communication between PCM-HMI applications on the same computer, in the local networks or on the Internet. This solution resolved the challenge related on middleware choice and separation of the industrial activity.

Sensing the weaknesses of the framework, the authors intend: to clearly define the vertical planes such as security, timing and management; to improve support for automatic configuration of fieldbuses; to directly connect the DDS object to the FCM in order to retrieve data from fieldbuses through the objects' dictionary (there is a risk of creating a security breach, because the same object has direct access to process data and may publish the data acquired from the sensors and transducers on the Internet and can take commands from the Internet for the actuators); to be embedded, even partially, based on a new profile, in TAO and DDS, the address space concept and the information model from OPC UA; to develop tools for the easy configuration of DDS objects; to develop OPC UA security concepts in OpenDDS.

ACKNOWLEDGMENT

This work was partially supported from the project “Integrated Centre for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for fabrication and control”, Contract No. 671/09.04.2015, Sectoral Operational Program for Increase of the Economic Competitiveness co-funded from the European Regional Development Fund.

REFERENCES

[1] K. Ashton, “Internet of Things,” RFID Journal, June 22 2009.

[2] Qazi Mamoon Ashraf, Mohamed Hadi Habaebi, „Autonomic schemes for threat mitigation in Internet of Things,” Journal of Network and Computer Applications, Volume 49, March 2015, Pages 112-127, ISSN 1084-8045, http://dx.doi.org/10.1016/j.jnca.2014.11.011.

[3] R. van Kranenburg, The Internet of Things: A Critique of Ambient Technology and the All-Seeing Network of RFID. Institute of Network Cultures, 2008

[4] Jordán Pascual Espada, Ronald R. Yager, Bin Guo, Internet of things: Smart things network and communication, Journal of Network and Computer Applications, Volume 42, June 2014, Pages 118-119, ISSN 1084-8045, http://dx.doi.org/10.1016/j.jnca.2014.03.003.

[5] Scott MacDonald, Whitney Rockley, McRock CAPITAL, The Industrial Internet of THINGS – IIoT Report, 2014.

[6] Roselli, L.; Mariotti, C.; Mezzanotte, P.; Alimenti, F.; Orecchini, G.; Virili, M.; Carvalho, N.B., "Review of the present technologies concurrently contributing to the implementation of the Internet of Things (IoT) paradigm: RFID, Green Electronics, WPT and Energy Harvesting," Wireless Sensors and Sensor Networks (WiSNet), 2015 IEEE Topical Conference on , vol., no., pp.1,3, 25-28 Jan. 2015.

[7] Bolic, M.; Rostamian, M.; Djuric, P.M., "Proximity Detection with RFID: A Step Toward the Internet of Things," Pervasive Computing, IEEE , vol.14, no.2, pp.70,76, Apr.-June 2015.

[8] Eugster, P.; Sundaram, V.; Xiangyu Zhang, "Debugging the Internet of Things: The Case of Wireless Sensor Networks," Software, IEEE, vol.32, no.1, pp.38,49, Jan.-Feb. 2015.

[9] Senouci, Mustapha Reda, et al. "WSNs deployment framework based on the theory of belief functions." Computer Networks 88 (2015): 12-26..

[10] Palattella, M.R.; Accettura, N.; Vilajosana, X.; Watteyne, T.; Grieco, L.A.; Boggia, G.; Dohler, M., "Standardized Protocol Stack for the Internet of (Important) Things," Communications Surveys & Tutorials, IEEE , vol.15, no.3, pp.1389,1406, Third Quarter 2013, doi: 10.1109/SURV.2012.111412.00158.

[11] Sanchez, Luis, et al. "SmartSantander: IoT experimentation over a smart city testbed." Computer Networks 61 (2014): 217-238.

[12] Jian An, Xiaolin Gui, Wendong Zhang, Jinhua Jiang, Jianwei Yang, Research on social relations cognitive model of mobile nodes in Internet of Things, Journal of Network and Computer Applications, Volume 36, Issue 2, March 2013, Pages 799-810, ISSN 1084-8045, http://dx.doi.org/10.1016/j.jnca.2012.12.004.

[13] Zheng Yan, Peng Zhang, Athanasios V. Vasilakos, A survey on trust management for Internet of Things, Journal of Network and Computer Applications, Volume 42, June 2014, Pages 120-134, ISSN 1084-8045, http://dx.doi.org/10.1016/j.jnca.2014.01.014.

[14] Satyanarayanan, M.; Simoens, P.; Yu Xiao; Pillai, P.; Zhuo Chen; Kiryong Ha; Wenlu Hu; Amos, B., "Edge Analytics in the Internet of Things," Pervasive Computing, IEEE , vol.14, no.2, pp.24,31, Apr.-June 2015, doi: 10.1109/MPRV.2015.32

[15] Therese Sullivan, The Cutting-Edge of IoT, How does the IoT really change the future of commercial building operations?, November 2014,AutomatedBuildings.com, November 2014, http://www.automatedbuildings.com/news/nov14/articles/buildingcontext/141030095606bldgcntx.html

[16] Akram Hakiri, Pascal Berthoua, Aniruddha Gokhale, Douglas C. Schmidt, Gayraud Thierry, Supporting End-to-end Scalability and Real-time Event Dissemination in the OMG Data Distribution Service over Wide Area Networks , Elsevier Journal of Systems and Software, 2013.

[17] D. C. Schmidt, A. Corsaro, and H. V. Hag. 2008. Addressing the challenges of tactical information management in net-centric systems with DDS. Journal of Defense Software Engineering, 24–29.

[18] OMG. 2007. Data Distribution Service for Real-Time Systems. v1.2.

[19] http://www.omg.org/spec/DDS/1.2/

[20] Li Da Xu, Wu He, Shancang Li, Internet of Things in Industries: A Survey, DOI 10.1109/TII.2014.2300753, IEEE Transactions on Industrial Informatics, 2014.

[21] Q. Wei, S. Zhu, C. Du, “Study on key technologies of Internet of Things perceiving mine,” Procedia Engineering, vol.26, pp.2326-2333, 2011.

[22] Bo Cheng, Xin Cheng, Junliang Chen, Lightweight monitoring and control system for coal mine safety using REST style, ISA Transactions, In Press, Corrected Proof, Available online 8 August 2014.

[23] ACATECH – Recommandations for implementing the strategic initiative INDUSTRIE 4.0. April 2013. http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf

[24] IoT@Work, https://www.iot-at-work.eu/ (Accessed April 2016).

[25] Herman Storey (co - chair ISA 100), Rick Bullota and Daniel Drolet. The Industrial Internet of Things, http://www.csemag.com/single-article/the-industrial-internet-of-things/c98837a0efec387df9fc14c2d e0a3b2f .html (Accessed April 2016).

[26] Ovidiu Vermesan, Peter Friess, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, pp158, ISBN: 978-87-92982-73-5, River Publishers, 2013.

[27] International Telecommunications Union, ITU-T Y.2060, Overview of the Internet of things, 2012.

[28] OMG, Data Distribution Service (DDS) http://www.omg.org/hot-topics/dds.htm (Accessed April 2016).

[29] Vasile-Gheorghita Gaitan, Nicoleta-Cristina Gaitan, Ioan Ungurean, A flexible acquisition cycle for incompletely defined fieldbus protocols, ISA Transaction journal, Elsevier, Volume 53, Issue 3, pp. 776-786, May 2014.

[30] Yucel Cetinceviz, Ramazan Bayindir, Design and implementation of an Internet based effective controlling and monitoring system with wireless fieldbus communications technologies for process automation—An experimental study, ISA Transactions journal, Elsevier, Volume 51, Issue 3Pages 461–470, May 2012.

[31] H. Perez, J.J. Gutierrez, "A survey on Standards for real-time distribution middleware" Journal ACM Computing Surveys, vol. 46, issue 4, March 2014, article no.49.

[32] J. Bard and V. J. Kovarik. 2007. Software Defined Radio: The Software Communications Architecture. Wiley-Blackwell. ISBN: 0-47086-518-0.

[33] M. Amoretti, S. Caselli, and M. Reggiani. 2006. Designing distributed, component-based systems for industrial robotic applications. In Industrial Robotics: Programming, Simulation and Applications, Low Kin Huat (Ed.). ISBN: 3-86611-286-6, InTech, DOI:10.5772/4892.

[34] OMG. 2012. Corba Core Specification. v3.3. http://www.omg.org/spec/CORBA/3.3/, or http://www.omg.org/spec/ZIOP/ (Accessed April 2016).

[35] OMG. 2005. Realtime Corba Specification. v1.2. http://www.omg.org/spec/RT/1.2/ (Accessed April 2016).

[36] D. C. Schmidt. 2005. TAO Developer's Guide: Building a Standard in Performance. Object Computing, Inc.

[37] M. Ryll and S Ratchev. 2008. "Application of the data distribution service for flexible manufacturing automation." International Journal of Aerospace and Mechanical Engineering 2, 3, 193–200.

[38] M. Gillen, J. Loyall, K. Z. Haigh, R. Walsh, C. Partridge, G. Lauer, and T. Strayer. 2012. Information dissemination in disadvantaged wireless communications using a data dissemination service and content data network. In Proceedings of the SPIE Conference on Defense Transformation and Net-Centric Systems, Vol. 8405.

[39] OMG. 2009. The Real-Time Publish-Subscribe Wire Protocol. DDS interoperability wire protocol specification. v2.1. http://www.omg.org/spec/DDSI/2.1/

[40] OMG. 2012. Extensible and Dynamic Topic Types for DDS. v1.0. http://www.omg.org/spec/DDS-XTypes/1.0/ (Accessed April 2016).

[41] H. P´erez, J. J. Guti´errez, and M. Harbour. 2012. Adapting the end-to-end flow model for distributed Ada to the Ravenscar profile. Ada Letters 33, 1, 53–63.

[42] http://www.omg.org/spec/DAIS/1.1/PDF (Accessed April 2016).

[43] http://www.omg.org/cgi-bin/doc?omg/11-08-01.pdf (Accessed April 2016).

[44] https://www.rti.com/docs/RTI_Security_Solutions.pdf (Accessed April 2016).

[45] http://www.prismtech.com/opensplice/resources/documentation, OpenSplice_SecurityConfiguration_Guide_A131. Pdf (Accessed April 2016).

[46] https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-2-security-model/ (Accessed April 2016).