An Event-Triggered Smart Sensor Network Architecture

Eríco Meneses Leão, Luiz Affonso Guedes, and Francisco Vasques, Member, IEEE,

Abstract—A smart transducer is the integration of a sensor/actuator element, a processing unit and a network interface. Smart sensor networks are composed of smart transducer nodes interconnected through a communication network. This paper proposes a new architecture for smart sensor networks, that is driven by events (asynchronous data). The events are derived from a data compression algorithm embedded in the smart sensor, which compresses data from the sensor. The proposed architecture also provides configuration and monitoring data to manage the distributed system.

I. INTRODUCTION

From a systemic approach, industrial automation can be characterized as a set of techniques that enable the construction of active subsystems. These subsystems must have the capability to interact with the industrial processes for control, monitoring, and supervision proposes. Many activities in industrial automation have real time requirements.

The evolution experienced by the industrial automation area have always been closely associated with the technologies developed and supported by commercial suppliers. In the last few decades, we have been evolving from a strongly centralized technology to an essentially distributed technology. Within this new approach, components are interconnected by digital communication networks. Thus, the traditional sensors based in the 4-20mA standard are being replaced by digital devices. These devices may have sensors, actuators, and control functionalities and are endowed of digital processors and communication systems. Within this context, a smart sensor is defined as the integration of an analog or digital sensor or an actuator element, a processing unit, and a network interface [1]. In this way, modern architectures of industrial automation are characterized for using a set of the smart transducers, usually connected through communication network with real time proprieties.

The distributed approach provides a significant improvement in the flexibility and scalability aspects of the industrial processes; however, it also brought new scientific and technological challenges, such as new models and algorithms for real time and safe communications under costs and environmental restrictions.

Recently, the first standards for smart sensor have been introduced. Among them, we highlight as the most important the IEEE 1451 [2] and the OMG’s Smart transducers standards [3]. The IEEE 1451 standard is more concerned with the communication interfaces between a standard sensor and the processing module, and between this module and the communication network. However, it does not impose the use of any communication network in particular. On the other hand, the OMG’s standard defines a master-slave architecture, whose elements communicate through a time-triggered synchronous communication service. OMG devices are more simple than IEEE ones, but they have a very well-defined interface using CORBA (Common Object Request Broker Architecture).

The objective of this paper is to present a distributed smart sensor network architecture based on the OMG standard. However, differently to the original OMG proposal, the proposed approach uses an asynchronous event-triggered mechanism to transmit the data from the smart sensors. This asynchronous behavior is a consequence of the implementation of an embedded data compression algorithm in the smart sensor, which will send only the relevant data from the raw data mass. Thus, the proposed approach saves an important amount of communication bandwidth.

The rest of the paper is organized as follows: Section II introduces the properties of the smart sensor networks and describes the smart transducer interface standard adopted by the OMG. Section III depicts the proposed smart sensor architecture and its features. In section IV a case study related to the implementation of the compression algorithm over real sensor data is presented. The paper is concluded in section V.

II. SMART SENSOR NETWORKS

Transducer is a general designation for a sensor or actuator, which are the detection and actuation devices in a determined process, respectively. These devices have relevant roles within the industrial automation context.

With the advent of the modern microcontrollers and the large availability of tools and resources for the processing of digital systems, it became feasible the introduction of a high computation capacity into digital transducers [4]. A smart transducer is the integration of an analog or digital sensor or actuator element, a processing unit, and a communication interface [5].

The smart sensor design deals not only with interchanges among devices, but also with interoperability and availability of information in real time. The networked operation of the smart sensor using standardized interfaces allows sharing information and resources. Thus, it allows the integration of all the process for control and supervision proposes, for example.

The most critical factor for the design of a smart transducer is the design of the interfaces to a standardized smart transducer. A smart transducer interface must be conform to
one of the world-wide standards. Such a standard for a real-time communication network has been long sought, but efforts to find one agreed standard have been hampered by vendors, which were reluctant to support a single common standard in fear of losing some of their competitive advantages [6].

Suppose that sensors and actuator are implemented as smart transducers with a network interface in a distributed system. It is possible to connect these smart transducers to a communication system with broadcast characteristics [1]. The following requirements for smart transducers have been identified [1]:

- **Real-time operation**: Most applications for smart transducer require timely actions.
- **Complexity management**: The smart transducer should provide means to manage the system complexity when composing or changing a network of transducers.
- **Maintenance support**: Systems that are in operation for an extended period of time usually require maintenance access to smart transducers.
- **Deterministic behavior**: A system is deterministic if a given set of inputs always lead to the same system output.

The user of a smart transducer service can access its data via an abstract interface that hides the internal complexity of the transducer. The transducer manufacturer will deal with instrumenting the local transducer and signal conditioning in order to export the transducer’s service in a standardized way [5].

A distributed system with a large numbers of smart transducers should provide a generic approach for automatic configuration. This plug and play characteristic saves time and therefore leads to better maintainability and lower costs. It also requires less exigency for the qualification of the personnel responsible for the configuration [7].

**A. OMG Smart Transducer Standard**

In December 2000, the Object Management Group (OMG) called for a proposal of a smart transducer interface standard. In response, a new standard has been proposed that comprises a time-triggered transport service within the distributed smart transducer network. It also defines a interface to a CORBA environment, which enables maintenance activities [1], [6]. This smart transducer standard has been adopted by the OMG in January 2002 [3].

The OMG standard defines a set of clusters with up to 250 smart transducer node connected to a bus. Each cluster is connected to a CORBA gateway via a master node, which is responsible for controlling the communication within the cluster. There may be redundant shadow masters to support fault tolerance. The structures of the OMG smart transducer architecture is show in the Figure 1.

**B. OMG Smart Transducer Interfaces**

Access to the smart transducer data is achieved by assigning three different interfaces to each smart transducer node [3], [1]:

- **DM interface**: Diagnostic and management interface. It is used by the master node to read smart transducer records, to parameterize and calibrate sensors or to collect diagnostic information for maintenance purposes.
- **CP interface**: Configuration and planning interface. It is used by the master node to integrate and setup newly connected nodes.
- **RS interface**: Real-time service interface. It establishes a periodic communication with predictable timing behavior among the smart transducer nodes.

The transfer of information between a smart transducer and its client is achieved by sharing information contained in an internal interface file system (IFS), which is encapsulated in each smart transducer. Thus, the IFS is the source and sink for all communication activities [6].

**III. PROPOSED ARCHITECTURE**

The proposed smart sensor architecture is based on the OMG’s standard. The choice for this standard was motivated by its simple and well defined data access interface. However, differently to the OMG standard, the proposed architecture follows an event-triggered approach. That is, the smart sensors send through the network only the relevant information about the process. The transfer of this information is triggered by asynchronous events.

The event-triggered approach follows the “on demand” paradigm. In this approach, it is harder to guarantee the temporal requirements than in the time-triggered approach. However, it presents good performance in applications with sporadic actions and in best-effort real time systems with high level of resource utilization [1].

Thus, the event-triggered approach is indicated to environments characterized by asynchronous activities. In these cases, an application would have to send data through the communication network only when a relevant event has been generated.

Figure 2 shows the basic components of the proposed architecture. This architecture is composed by clusters. In turn, each cluster can have one master node and up to 255...
slave nodes, which are interconnected by a field network. The master node is a most powerful processing device and it is designed to manage the cluster. It can also communicate with other master nodes through a supervision network. Thus, it is the responsible for management, control, and configuration activities throughout the system.

A. Proposed Master Node Structure

The master node has two database types: a real time database (RTDB) and a traditional database (DB). The RTDB has to guarantee the temporal deadlines of its transactions. Thus, some data will be valid just for a specific time interval [8]. The master node records in its databases the most relevant information from the smart sensors. These databases are accessed through the diagnostic and maintenance interface (DM) using the supervision network. The master node accesses information from the slave nodes using the RT client (real time client) and the CP client (configuration and planning client).

B. Field Network

In this paper, rather than to propose a new field network, we just define the services that a field network needs to provide. Amid them, we should highlight the following:

- **Clock Synchronization**: The field network must support a service that enables the synchronization of all clocks in the network.
- **Real-Time Communication Service**: The field network must support an adequate scheduling service to guarantee the deadlines of the transferred messages.
- **Fault Tolerance**: The field network must support a fault-tolerant communication service. By definition, a system is tolerant to faults if, even when a fault is verified, the system is capable of working, possibly in a degraded way [9].

C. Proposed Slave Node Structure

A smart transducer node requires two types of interfaces for accessing its data, a compression algorithm and a buffer to store temporary data to applying in data compression, as in Figure 3. The compression algorithm is responsible for selecting the relevant data information, generation an asynchronous flow of data this behavior leads to unpredictable timing intervals between consecutive data transfers from the sensor. However, the time of sending a datum is not totally unpredictable, due to the minimum and maximum time provided the compression algorithm to the transmission, as will be shown ahead. So, an event-triggered approach would be efficient for the communication of these data.

With the larger autonomy provided to the smart sensor, which will now send only relevant data, there is a significant reduction of the transferred data, resulting in a smaller bandwidth utilization. This way, it is possible to connect a larger amount of smart sensors to the network. However, they will become more complex and so will demand a larger processing capability. Nevertheless, with the growing technological progress it is possible to design low-cost smart sensors with high processing capabilities. Being so, the embedded compression algorithm is one of the research targets that must be addressed for the architectural network of event-triggered smart sensors.

Sensors require two types of interfaces to access the data (RS and CP), where the communication is supported by two different communication models (publisher-subscriber and client-server):

- **RS interface** - real-time service interface. It is used to transfer real-time data to the cluster. The data generated by the smart sensors will be published in the net and consumed by the functions of the system.
- **CP interface** - configuration and planning interface. Through this interface it is possible to identify new nodes connected to the network, to transfer new configuration parameters to the sensor, as values of compression deviation, maximum and minimum time for the compression algorithm, besides information as the identification of the sensor in the distributed system.

1) **Publisher-Subscriber Model**: The communication model used by the RS interface is the real-time publisher-subscriber model (RTPS). This model of exchanging data favors the message exchange with time parameters amid devices between two entities: the publisher, responsible by sending the messages, and the subscribers, responsible for consuming these messages, in case they interest them [10], [11]. The messages...
sent by the slave node through the RS interface will be consumed by the various functions of the system concerned by such data. These messages can be send as commands to the actuators or stored in databases in accordance with their requisites; they can be real-time databases (RTDB) or traditional databases (DB), as shown in Figure 4.

The data packets sent by smart sensors through the RS interface contain various fields with important information from the smart sensors, which will be consumed by the system functions to different finalities. For example, the `operation_time()` function will access the TIME field and will store this datum in the real-time database, in order to provide the supervision network the functioning time of a determined smart sensor. The `sensor_historic()` function will access the information in the DATA field, in order to make available the historic file of flags for a smart sensor. The packet’s TAG field is responsible for identifying the sensor; the TYPE field by the identification of the type of the packet to be transferred, while the QUALITY field will contain information about the quality flag of the datum (good, regular, bad, not determined). The CRC field is responsible for controlling the packet’s error. The structure of such packet is illustrated in Figure 5.

<table>
<thead>
<tr>
<th>TAG</th>
<th>TYPE</th>
<th>TIME</th>
<th>DATA</th>
<th>QUALITY</th>
<th>CRC</th>
</tr>
</thead>
</table>

Fig. 5. Structure of a slave node message

2) **Client-Server Model**: The CP interface is accessed directly by the master node, which can perform configuration activities, parameter passing and reconfiguration of new nodes (Figure 6). As an example of the configuration of a new node connected to the network through the CP interface, a master node through its function `new_device()` stays monitoring the system in order to search new nodes connected to the network. Thus, a new smart sensor will ask its inclusion in the system and will be given by the master node, through the function `configure()`, the necessary parameters to start functioning in the network. From this point onwards, after confirming the received parameter (`confirm()`), the sensor will start sending its data packets through the RS interface. Figure 7 illustrates this procedure, while Table I present some important system functions.

![Diagram](image.png)

**Fig. 6. Client-Server Model for Proposed Architecture**

**Fig. 7. Inclusion proceeding of a new node**

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>request_inclusion()</td>
<td>CP</td>
<td>slave node requisites to master node its inclusion in the network.</td>
</tr>
<tr>
<td>configure()</td>
<td>CP</td>
<td>master node sends configuration parameters of a new slave node.</td>
</tr>
<tr>
<td>reconfigure()</td>
<td>CP</td>
<td>master node sends a reconfiguration to a slave node.</td>
</tr>
<tr>
<td>stop()</td>
<td>CP</td>
<td>master node sends a stop request to a slave node.</td>
</tr>
<tr>
<td>start()</td>
<td>CP</td>
<td>master node authorizes the functioning of a slave node.</td>
</tr>
<tr>
<td>check_sensor()</td>
<td>CP</td>
<td>master node verifies if one or more slave nodes are active.</td>
</tr>
<tr>
<td>transmit()</td>
<td>RS</td>
<td>slave node sends a determined packet to the network.</td>
</tr>
<tr>
<td>operation_time()</td>
<td>RS</td>
<td>responsible for the time functioning of a smart sensor.</td>
</tr>
<tr>
<td>diagnostic()</td>
<td>RS</td>
<td>responsible by diagnostic information of a smart sensor.</td>
</tr>
</tbody>
</table>

**IV. CASE STUDY**

This section presents a case study using the **Swinging Door** [12] compression algorithm. The main goal of this implementation is to demonstrate that compression and filtering activities can be used in smart sensors, in order to send just the relevant data without losing information integrity.

The **Swinging Door** algorithm is classified as a compressor with information loss. The basic principle of the **Swinging Door** algorithm is to place the first datum marked as the starting point of the first compression interval. The final point
of this compression interval also actuates as final point of the next compression interval [13].

The algorithm contains three main parameters: the compression deviation, the minimum and the maximum compression time. The algorithm defines a parallelogram as shown in Figure 8. Starting from the initial point, if no received value exceeds the parallelogram until the final point, no intermediate point is stored. In this case, the final point will be stored, and a new starting point will be defined. If some intermediate value exceeds the parallelogram and this value is not synchronous with the minimum time scheduled, then the former value will be stored. The minimum time of compression serves to filter noisy signals.

For the first tests we used real data extracted from a real-time gas distribution monitoring system; specifically, we made an option for an outlet variable. For testing effects, a hypothetical situation was considered where the sensor marks were synchronous, generating a new value each second. Having in view a better visualization of the results, 5000 bank values have been used, from a total of 25994 registers. Table II describes some of the results obtained from practical experiments. It comes out that when the algorithm’s main parameters are varied, different compression rates are attainable. This way, one can reduce the data traffic nearly 10 times without losing relevant data.

![Swinging Door Algorithm](image)

**Fig. 8. Swinging Door Algorithm**

For this particular example, the range of compression between 40% and 90% presents a low variation in square mean error generated by the reconstruction of the compacted data also rises.

![Original Data of the Sensor](image)

**Fig. 9. Original Data of the Sensor**

![Data compression of 88.42%](image)

**Fig. 10. Data compression of 88.42%**

It is import to say that the *Swinging Door* algorithm depends on formerly defined parameters. Thus, the compression rates of the sensors’ values will be dependent on the master node’s established configuration. For example, a pressure sensor will have a parameter configuration different from that one of a temperature sensor.

Table II

<table>
<thead>
<tr>
<th>Deviation (%)</th>
<th>$t_{\text{min}}$ (s)</th>
<th>$t_{\text{max}}$ (s)</th>
<th>Compression (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>2</td>
<td>10</td>
<td>4.84</td>
</tr>
<tr>
<td>0.05</td>
<td>2</td>
<td>3</td>
<td>15.48</td>
</tr>
<tr>
<td>0.05</td>
<td>2</td>
<td>4</td>
<td>18.37</td>
</tr>
<tr>
<td>0.05</td>
<td>2</td>
<td>10</td>
<td>19.26</td>
</tr>
<tr>
<td>0.3</td>
<td>2</td>
<td>3</td>
<td>42.28</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>4</td>
<td>62.58</td>
</tr>
<tr>
<td>0.5</td>
<td>2</td>
<td>10</td>
<td>80.78</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>10</td>
<td>86.7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10</td>
<td>87.92</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>10</td>
<td>88.42</td>
</tr>
</tbody>
</table>

TABLE II

**COMPRESSION RATES OF THE EXPERIMENT**

For this particular example, the range of compression between 40% and 90% presents a low variation in square mean error generated by the reconstruction of the compacted data also rises.
error, which characterizes it as a good compression range to be used.

V. CONCLUSION

This paper proposes an event-triggered smart sensor network architecture. The main feature of the proposed architecture is the integration of a data compression algorithm into the smart sensors. This approach saves network bandwidth, because the sensors only send relevant data.

With the achieved results through the realized experiments, it is possible to conclude that the local compression process in each smart sensor can decrease considerably the data exchanges in the communication network. Further work will include the communication control in the event-triggered smart sensor architecture, using schedule techniques with priority to the messages for the purpose of avoiding feasible collisions and packet loss.

REFERENCES


Erico Leão received the BSc degree in computer science from the Federal University of Piauí, Brazil, in 2005. Currently, he is a MSc student in electrical engineering from the Federal University of Rio Grande do Norte, Brazil. His research interests include real-time communication systems, and real-time distributed system architectures.

Luiz Affonso Guedes has a graduate in electrical engineering from the Federal University of Pará, Brazil, in 1987, and PhD degree in Computer Engineering from State University of Campinas, Brazil, in 1999. Since 2003, he is an associate professor in the Department of Computer Engineering and Automation at Federal University of Rio Grande do Norte, Brazil. His main research interests include Communication System to Industrial Application and High Level Programming Languages to Industrial Automation.

Francisco Vasques (M’00) received the BSc degree in Electrical Engineering from the University of Porto, Portugal, in 1987 and both the MSc and PhD degrees in Computer Science from LAAS-CNRS, Toulouse, France, in 1992 and 1996. Since January 2004, he is Associate Professor at the Mechanical Engineering Department of the University of Porto, where he is at the head of the Master Programme on Mechanical Engineering.

Prof. Francisco Vasques is author or coauthor of more than 100 technical papers in the areas of real-time systems and industrial communication systems. His current research interests include real-time communication systems, industrial communications, fault-tolerant systems, and real-time system architectures. Since 2007, he is an Associate Editor of the IEEE Transactions on Industrial Informatics for the topic on Industrial Communications.