# Network Intrusion Detection System Embedded on a Smart Sensor

Francisco Maciá-Pérez, Francisco J. Mora-Gimeno, Diego Marcos-Jorquera,
Juan A. Gil-Martínez-Abarca, Héctor Ramos-Morillo, and Iren Lorenzo-Fonseca

***Abstract*—This paper proposes a *Network Intrusion Detection System* (NIDS) embedded in an *Smart Sensor* inspired device, under a Service Oriented Architecture (SOA) approach, able to operate independently as an *anomaly-based NIDS* or integrated, transparently, in a *Distributed Intrusion Detection System* (DIDS). The proposal is innovative, because it combines the advantages of *Smart Sensor* approach and the subsequent offering of the NIDS functionality as a service with the SOA use in order to achieve their integration with other DIDS components. The main goal of the work is to reduce the huge volume of management tasks inherent to this type of network services, as well as facilitating the design of DIDS whose managing complexity could be restricted within well defined margins. The work also addresses the construction of a physical sensor prototype. This prototype was used to carry out the tests that has demonstrated the proposal's validity, providing detection and performance ratios similar to those of existing IDS, but with the advantage of a zero-maintenance approach.**

***Index Terms*—Intrusion Detection Systems (IDS), Embedded Systems, Smart Sensors, Service Oriented Architectures (SOA).**

## I. Introduction

INTRUSION Detection Systems (IDS) are a vital element in the communications infrastructure of organizations and represent one of the main security tools for computer networks.

The role of the security administrator in intrusion detection is not easy; systems and services are becoming increasingly sophisticated and complex, as well as, consequently, their configurations, and new attacks and vulnerabilities are constantly arising. Moreover, and due to the undeniable success of networks as communications tool, the number of interconnected nodes and the volume of information to be managed are growing at an alarming rate.

Traditional IDS are based on low level attacks and generate isolated alerts, although there is a logical connection between them. Furthermore, the huge number of generated alerts becomes unmanageable and security administrators are unable to cope with them, making impossible to scrutinize and understand adequately the network's security status [1]. In order to solve this problem, the distributed intrusion detection systems (DIDS) combine all these scattered alerts and make use of their logic relationship, thus obtaining additional information.

DIDS are currently as necessary as complex, due to the fact that they involve several technologies, devices and network resources, as well as sophisticated management tasks which are beyond the scope of many users or organizations which do not have a highly specialized team of administrators.

There are still many open fronts in the field of intrusion detection, which are not solely concerned with improving detection ratios or with reducing the number of false positives that they generate. Some of them are: a) IT technological infrastructure which supports this type of system is increasingly sophisticated thus increasing both the complexity and number of associated management tasks; b) these systems are increasingly required to generate more information which overloads the network and the intrusion analysis systems themselves.

Of all the problems, these are the ones which our proposal addresses in seeking architectures for the effective distribution of system logic, reducing as far as possible the impact of increased network traffic, keeping detection levels of the present systems and proposing scalable solutions, easy to implement and with a zero-maintenance philosophy.

The huge range of small, low-cost embedded devices provided with one or more sensors, interconnected through wireless or cable networks integrated to the Internet, provide endless opportunities for monitoring and controlling organizations, homes, cities or the environment. Examples of this kind of devices are hardware probes RMON-based [2]. Furthermore, Smart Sensors technology gives support to specific requirements such as restrictions in the assignation of resources, compactness and flexibility to be adapted to various types of sensors, interfaces and computational communications and hardware [3]. These characteristics make the embedded devices in general and the smart sensors in particular an ideal framework for resolving many of the problems detected in the *Network IDS* (NIDS) [4], [5].

In view of the foregoing, this article proposes to apply the technology of *Smart Sensors* to design a physical device in which a NIDS capable of understanding the captured traffic and offering it on demand is embedded.

Manuscript received May 29, 2008. Accepted for publication May 18, 2010.



The authors are with the Department of Computer Science and Technology, University of Alicante, P.O. Box 99, 03080 Alicante, Spain (email:pmacia@dtic.ua.es).

Although the proposal shall be developed more extensively in subsequent sections we may summarize it for the time being as follows: design of small network devices to act following the principles of intelligent network sensors, capable not only of monitoring network traffic but also of processing it online. They will search for anomalies caused by intrusion attempts and generate alerts in the event of such anomalies, as well as storing and communicating the aforementioned alerts as required. In practice, the network sensor behaves like a NIDS for detecting anomalies acting with a *Smart Sensor* philosophy. The main advantage of these tiny, self-managed devices is that they can be incorporated to DIDS without noticeably increasing the global complexity of the system.

Taking this idea, in the upcoming section an overview of previous works and results in this field is provided. Following that, the proposal for a *Smart Sensor* is explained together with the general development scenario in which it could act as part of a DIDS; then, the design of the functional prototype is revised in order to make the relevant tests and this has been used to validate the proposal; finally the main conclusions are presented based on the work carried out as well as research prospects.

## II. Background

The main challenge for present day IDS is to be able to detect new attacks based on other previously observed events. IDS research, based on anomalies has led to the use of various techniques for modeling normal behavior and carrying out the process of analysis [6]. The most frequently used tool is statistical analysis where the normal model comprises a group of statistical variables. For example, a linear combination of six parameters [7] or a measure based on distribution of characters in order to detect anomalies in the content of the network packets [8].

Nevertheless, there are many anomaly detection systems which use machine-learning techniques to gain knowledge by training the parameters that model the system's normal behavior. Neural networks are the most commonly used learning tools in IDS. Thus, we find their application in the detection of anomalies in programs based on system calls [9], in network protocols based on different session variables [10], [11], and on the key-words based application layer [12]. There are also approaches which combine neural networks in a hybrid IDS consisting on misused and anomalies.

Most of the works on IDS based on neural networks have used *Self Organizing Maps* (SOM) [13], [10], [14], [15] as they are easy to implement and due to the fact that they have the advantage of a lower training time as the network is unsupervised. Also, one of the more important characteristics of neural networks is that they can be implemented in hardware, making the most of the advantages of their inherent parallelism capabilities. In this sense, we can find a lot of related works: a neural chip implanted in a scalable platform [16], neural networks in DSP for robot positioning [17] and even a neural network FPGA-based coprocessor [18], [19].

The expansion of networks has rendered conventional IDS insufficient. In order to mitigate these deficiencies, distributed intrusion detection systems have been developed as a set of disseminated sensors which collaborate in detection tasks. However, current DIDS, built under a generally hierarchic architecture, display a lack of scalability that makes the use of decentralized techniques mandatory [20]. Decentralized approaches are currently considered the most suitable [21], to the point that in the detection process should only be involved the nodes where the intrusion occurs, acting in close cooperation with its nearby peers [22]. Furthermore, the use of a substantial number of sensors collaborating, together with the volume of information they generate and the growing speed of networks, hinders analysis and increases costs, making the use of light, autonomous detection-capable hardware mechanisms more and more appropriate [23].

Considerable breakthroughs have been made during the last decade in the field of technologies for the development of small network devices endowed with a more than acceptable computational capacity, autonomous function [24] and the possibility of embedding intelligence in them. Although until recently the cost of these devices did not justify their large scale incorporation to certain tasks and services management, this scenario has changed: the present trend toward the miniaturization of lucrative devices with astonishing computation and communication capabilities lays the foundations for proposals providing specific network services with minimal attention and administration requirements. These proposals are based on self configuration and management models compatible with service oriented architectures (SOA) and standardized protocols (UDDI, SOAP, uPnP), all embedded in a very small network device and at extremely reduced cost [25].

The huge range of small, low cost embedded devices provided with one or more sensors [2], [26], [27], interconnected through wired or wireless networks integrated into the Internet, opens a field of endless opportunities for monitoring and controlling our homes, cities or even the environment.

With the present technology, sensors are able to measure a wide range of magnitudes covering a broad spectrum of domains [28]. Furthermore, *Smart Sensors* usually integrate knowledge and constitute one of the key technologies for the future [3], [29].

Due to current restrictions on energy supply systems, reduced energy consumption levels has become a basic requirement for *Smart Sensors*. In order to improve the energy availability of the *Smart Sensors* it is possible to use protocols which provide the devices with energy through their communications network, such as the standard *Power over Ethernet* for wired networks [30] and WISA for wireless networks [31]. In general, *Smart Sensors* should make possible a configurable adjustment of the compromise between useful life (from an energy perspective) and the system's latency and effectiveness [29].

In addition, *Smart Sensors* and their applications have special requirements such as the need to save energy,

restrictions in the assignation of resources (CPU, memory, storage and bandwidth), compactness and flexibility to be adapted to various types of sensors, interfaces and computational communications and hardware [3].

The inability of traditional models of operating systems (OS) to match such requirements has led to the development of specific OS for *Smart Sensors* [3]. This is the case of TinyOS, an OS based on free software and open source for *Smart Sensors*, developed by the University of Berkeley which set the ground for numerous projects involving *Smart Sensor* networks. Other approaches more embedded device-oriented that may be useful are, for instance: ThreadX, a real-time OS for embedded development; and µCLinux, a Linux-based OS for embedded devices without memory handling units.

In order to achieve full integration, each *Smart Sensor* should go beyond ad hoc communication protocols, incorporating models, architectures and technologies able to offer their functionality in an open and standardized manner to facilitate interoperability between the sensors and the applications which make use of them. In this respect, service oriented architectures provide a model in which each network node offers its functionality through independent services accessible in a standardized manner [32]. SOA implies that all the different elements that form the system are related with each other as services. This service orientation is the next evolutionary step from the traditional client-server model with *Service Providers* instead of servers and *Service Consumers* instead of clients, but decentralizing the whole process by introducing the *Service Broker* as a new element, whose function is to keep the record of all services offered throughout the system and describe its use.

There are currently a set of standard technologies which make SOA based applications possible: XML language and WSDL languages to describe services, SOAP to provide services and UDDI services to register them [32]. The conjunction of these technologies constitutes the basis of Web Services (WS). Although other SOA implementation technologies like WS-* and DPWS are in the market, we opted for WS-I Basic Profile 1.0 version since its simplicity is a perfect match for our purposes requirements.

On the basis of this brief overview, it may be concluded that there are still issues remaining in terms of scalability, integrity, consumption and management of data volumes within the scope of IDS. It is also clear that IDS of anomalies based on SOM networks may serve as a reference for validating our proposal, due to its simplicity and to the fact that we are not interested in validating aspects of effective detection but rather efficiency in terms of resource management. Therefore it is clear that embedded systems and more specifically *Smart Sensors* may constitute a physical support, ideal for embedding functionalities of this type of IDS, mitigating to a large extent many of the problems detected.

## III. Global Scenario

Although the design of smart sensors via the implantation of IDS in a device to make it behave as an intelligent sensor is the core idea of this paper, it is also true that most of this proposal advantages make sense only when one or more of these sensors act forming a DIDS (under a SOA approach) as part of a much more sophisticated system. That's why we begin our proposal description by defining the general framework within smart sensors are integrated.

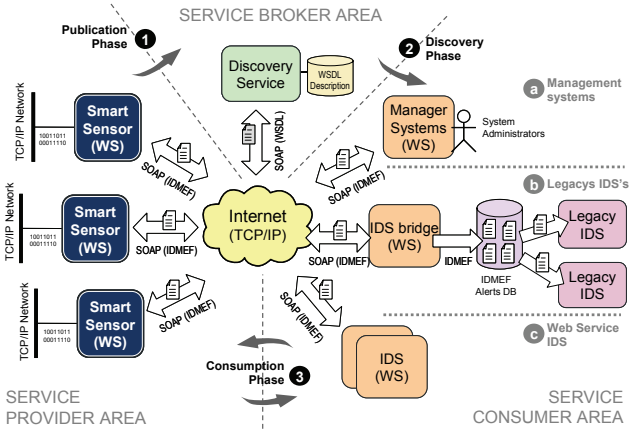

Fig. 1. Global scenario of the distributed IDS SOA-based proposal.

In Fig. 1 a typical scenario containing a DIDS whose components are structured under a SOA model is shown. In this scenario our embedded NIDS are seen as *Smart Sensors* and play the role of *Service Providers* offering *Web Services*. For this same reason the scenario has been divided into the three classical SOA scenario areas: *Service Provider Area*, *Service Broker Area* and *Service Consumer Area*.

The *Service Provider Area* may find different intelligent network sensors which act as NIDS and which have been connected to the networks to be investigated. The *Service Broker Area* contains the register servers responsible for maintaining information on the various services available throughout the system. The *Service Consumer Area* is the most heterogeneous and could be divided into three client's categories: (a) Management systems and remote terminals used by network or security administrators to access management tasks, or simply to compile information directly on analyzed traffic or alerts generated by a specific NIDS. These could range from simple Web navigators to complex management systems. (b) The second type of client comprise gateways which act as adaptors between our sensors, working under the SOA approach and other components or systems inherited from a traditional DIDS which do not support this approach. (c) Finally, there is a third type of client comprising components of a DIDS which incorporate WS technology in a native manner and, therefore, able to locate and contact *Smart Sensors* directly.

From a functional perspective, the scenario may also be described on the basis of different phases of the SOA model: *publication, discovery* and *consumption*.

1) *Publication* phase correspond with the first sensors state (when they are connected to the network). In this phase the sensor should locate a publication server using the UDDI protocol to send all the documentation describing

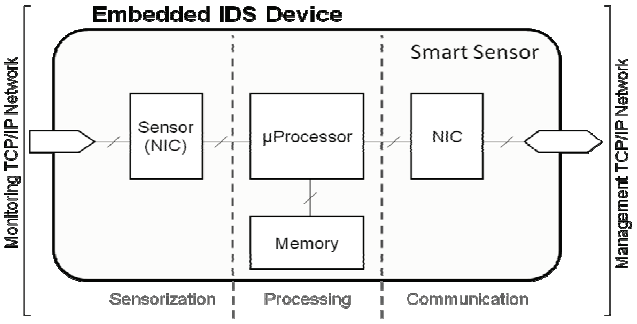


Fig. 2. Physical structure of the smart sensor which acts as an embedded device which serves as support for the sensor and the NIDS.

the service in the form of WSDL pages.

2) During the *discovery phase* any WS client wishing to consume an intrusion detection service offered by a S*mart Sensor* should know it previously. For this purpose firstly they will locate a registered service and will request sensor documentation in order to ascertain all its address details, how it should be approached and the manner in which the requested service will be returned to them. Once again the UDDI protocol will be used and the information will be based on WSDL pages.
3) *Consumption* is the most important phase of all as this is what grants real significance to the whole system. Clients in this phase, having discovered the available services, are in a position to consume them or to approach the *Smart Sensors* and request the services they offer directly. Communication between services and consumers in this phase shall be made through SOAP requests and responses.

## IV. Embedded IDS Proposal

Once established the general scenario, this section will focus on the design of the IDS embedded into a device inspired in *Smart Sensors*. Before starting up, it is important to clarify the existing relationship between our proposal and the traditional approach based on smart sensors networks (SSN).

In the field of SSN each sensor's autonomy is taken for granted. To achieve that, it is been considered, generally speaking, that smart sensors cannot be wired. Due to that, sensors should be efficient and collaborate in order to transmit the information to its final destination in the most sensible way consumption-wise.

In our proposal, the smart sensors-based device employed are twisted pair network sensors, so any limitation to their power autonomy becomes meaningless by the fact that they could be energized by an array of different techniques like Power over Ethernet and external power supplies, to mention a couple. By similar reasons there is no point in addressing the typical problems related to signal-retransmission collaboration and the like.

These are just a few SSN's characteristics, but there are others of similar importance. Among them, the way they were conceived, in which traditional sensors acting as mere transducers are bestowed with processor, memory and a communications module that allow the inclusion of know-how, information storage, and the supply-on-demand of the often processed-into-knowledge information.

These are the features that have guided our IDS design, embedding it into a smart sensor. In compliance to that, our network sensor is not only capable now of capturing the network traffic (taking the chance to solve its energy problems), but also of filtering and processing it in search, although primitively, of intrusion attempts and of offering its detection services (on-demand in this case) as well.

That is not to say that the sensor should be continuously consulted via pooling techniques, but, on the contrary, that it will just notify any intrusion attempt asynchronously, on demand and under the conditions and restrictions (detection thresholds, bandwidth, etc.) indicated.

Therefore, we can say that the fundamental basis of our proposal is the design of an intrusion detection system embedded in a miniaturized network device which will offer functionality as a Web Service. It will be provided with Service Oriented Architectures that in addition to act as a network sensor incorporated to a DIDS or to the system's administrators, will make possible that the IP traffic captured by them could be filtered and processed in the form of alerts.

For the alerts an intrusion detection message exchange format (IDMEF) will be used, defined by the IETF [33] in the RFC4765. This standard defines the data formats and the exchange procedures for sharing data of interest between IDS response systems and management systems which are required to interact together.

IDMEF alerts may be transmitted both continuously and on demand, when actually required, or whenever network resources are able to support the transfer without any significant hindrance to their performance assisting the design of scalable DIDS.

From a physical perspective the system has been designed as a *Smart Sensor.* Fig. 2 shows the physical structure of the sensor indicating its main elements: (a) the sensor itself which is formed by a network adapter for connection to the local area network (LAN) whose traffic is being analyzed. This adaptor should support promiscuous mode in order to capture all the network traffic. This element provides the system with sensitivity; (b) a microprocessor with embedded additional functionality in order to analyze the captured traffic online, looking for behavior patterns which could be considered anomalous, such as intrusion attempts, which are to be converted into alerts; (c) in addition to computing power, the sensor has been provided with a non-volatile internal memory to store the alerts generated by the IDS, together with the involved traffic; (d) finally, the device has an additional communications model for its connection to the management network (generally the Internet) by means of which it will receive analysis requests on possible intrusion attempts from other IDS subsystems, or administrative requests from the administration systems. The fact of having two independent network interfaces for monitoring and for communication, in

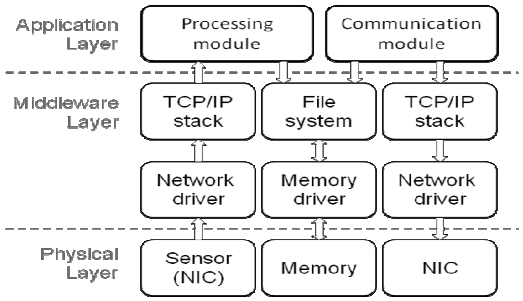


Fig. 3. Software architecture of the embedded IDS device.

addition to generalizing the proposal, makes it more viable in terms of a hostile, resources-starving environment like the world of computer network management.

Although from a physical perspective the IDS network system could be considered a S*mart Sensor,* from a functional viewpoint it could be labeled as a *service* for network intrusion detection organized into different layers: physical, middleware and application layer (see Fig. 3).

The *physical layer* considers the physical resources of the device from a functional point of view, in order to achieve its objectives the following elements should be connected, i.e. the local area network from which information is to be obtained, the internal memory of the device in which the processed information will be stored (i.e. IDMEF alerts generated) and the management network.

The *middleware layer* houses the modules which provide access to the basic resources of the physical layer (network adaptors, memories) encapsulating them and providing the upper layer with a standardized vision, free from physical details. According to the foregoing, the main blocks proposed are those of network and disk I/O management, together with a simple file system to facilitate non-volatile memory management and an implementation of a TCP/IP stack essential for all the processes of the application layer.

The *application layer* is the most important layer from a functional perspective. It contains the main functional components of the device. These components have been grouped into two large modules: a *processing module* and a *communications module.* These are analyzed in further detail below.

The *processing module* (see Fig. 4) represents the system core and is responsible for monitoring network traffic, its subsequent analysis and the storage of alerts detected along with the traffic and context data from which they originated. The processing module has in turn a *pre-processing filter* which adapts the traffic from the network, normalizing and

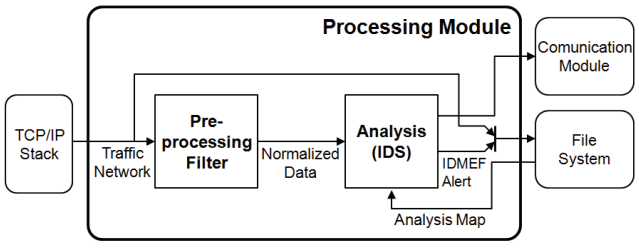


Fig. 4. Processing module main components.

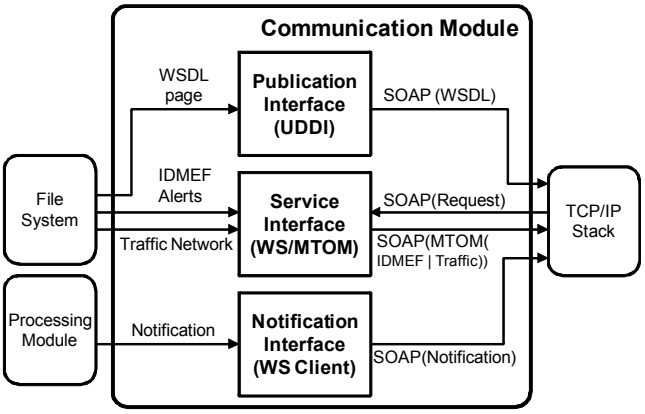


Fig. 5. Communication module main components.

converting it into an information pattern. This pattern is the input source of the *analysis module* which detects intrusions and generates alerts in IDMEF format. This module compares the input patterns with a map of patterns, previously stored in the non volatile memory of the device and, in the event of detecting an anomalous behavior it will catalogue it as an intrusion attempt and will store an alert indicating this fact, together with the traffic from which it originated and other appropriate information (such as the date and time it occurred or the type of traffic affected). The *analysis module* is a key element of the system since it implements the intrusion detection system and constitutes an anomaly-based network IDS.

The *communication module* (see Fig. 5) comprises the interfaces used by the device for external communications. The sensor provides three clearly differentiated interfaces: *publication interface*, *notification interface* and *service interface*.

The *publication interface* permits a registered server to be located and publish the services offered expressed on WSDL pages. To do so it will use, following a SOA pattern, the UDDI publication protocol on the SOAP application protocol.

The *service interface* is responsible for offering synchronically the device functionalities as Web Services, meaning: (a) to provide information about the alerts as well as about the analyzed network traffic and (b) to activate the sending of alert notifications asynchronously.

The *notification interface* is responsible for the asynchronous notification of the alerts occurrence as a function of the parameters (detection thresholds, bandwidths, response times, etc) defined via *service interface*.

## V. Prototype Implementation

The construction of the prototype of *Smart Sensor* that acts as an embedded IDS device has been divided in three well differentiated sections: the physical hardware support, the software applications and the intrusion detection system.

### A. Physical support

For the physical device a MOXA UC-7110-LX model network industrial device was used due to its distinctive features: small size, low consumption level and two network


Fig. 6. MOXA Device used as physical support to the prototype implementation.

interfaces (Fig. 6).

### B. *Software applications*

These devices incorporate the operating system µClinux v2.4.22. This has eased the development of the different applications by means of the usage of languages and well-known environments as the GCC compiler. At the same time, many open code developments were employed in order to implement the support libraries of the middleware layers; for instance: gSOAP v2.7 as application server. Fig. 7 shows the software architecture with the different modules that were implemented or used.

The MTOM protocol, supported by gSOAP, was used to transmit the network traffic associated with the generated alerts.

Since we are talking about an IDS security application, it is paramount to consider the security aspects of the system itself. In this sense the choice of gSOAP as applications server contributes with three important features: it allows us to use Message Transmission Optimization Mechanism (MTOM) to transfer the alerts and its associated traffic as opposed to other systems like FTP which introduce additional security holes, and it incorporates encrypted communications via SSL and expedites the use of protocols like HTTPS.

### C. *IDS*

The analysis module is the most important one in the application layer. For it an anomaly-based NIDS has been designed. Since a more efficient artificial neural network (ANN) for the intrusion detection is not intended to be tested in this work, a SOM has been chosen as detection engine due to its simplicity and widespread use in works of this kind. The proposed NIDS detects attacks at TCP connection level; to do so the parameters that will characterize each connection should be defined. These parameters represent the inputs to the ANN. For this purpose, a similar approach to the one exposed in [9] and [10] has been followed, selecting the five next characteristics: duration of the connection (DOC), number of bytes sent from the client to the server (SRC), number of bytes sent from the server to the client (DST), average of packets per second sent from the client (SPA) and average of packets per second sent from the server (DPA).

In order to obtain these data it is necessary to pre-process the incoming packets read from the system network interface. A RAW socket with a network interface in promiscuous mode has been used to obtain these packages. Firstly, since an ANN is to be built for each service in our system, data will be filtered to obtain just those packets that are addressed to the corresponding service. Secondly, the parameters that characterize each connection are obtained. For that, a module that constantly reads all the packets that form part of the different connections has been implemented to reconstruct the whole connection and, subsequently, to calculate the statistical variables of the incoming packets. Finally, in order to ensure that none of the five parameters predominates over the others due to the its values different dimensions, the incoming parameters must be normalized. Variance normalization has been chosen, so that each dimension of the input vectors has a variance of one. This normalization process is carried out keeping in mind the mean and the standard deviation of each previously calculated dimension. An IDS method description, in which our work has been based on, has been discussed in further detail in [11].

The SOM neural network uses the five components vectors obtained in the *pre-processing filter*. The ANN will classify each connection as either *normal* or *anomalous*, storing the IDMEF formatted result [33] in the internal memory of the device, for its later analysis and generating an alarm in case of anomalous connections.

## VI. Testing and Validation

To validate this proposal it has been developed a whole DIDS scenario based on SOA, as well as an IDS prototype embedded into a network device acting as a smart sensor.

### A. *Test scenario*

Following the scenario proposed in Fig. 1, to carry out the tests we have deployed two real computer networks: a management network and a working network.

The management network follows a SOA pattern, so we have grouped the interconnected network devices required to deploy all its functionality, into three distinct areas: *Service*

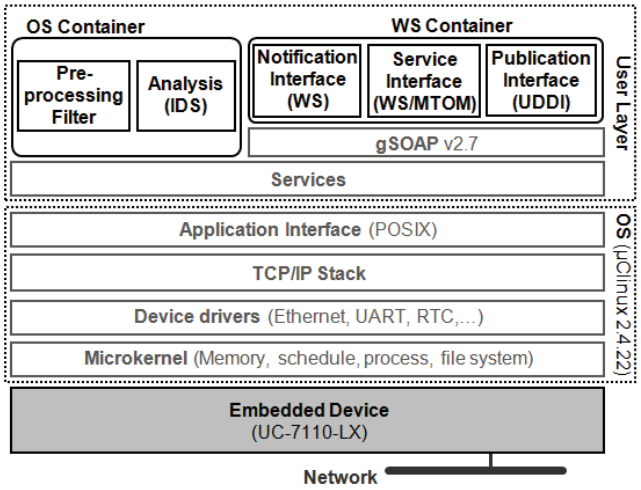


Fig. 7. Software architecture of the prototype depicting the various modules implemented or used.

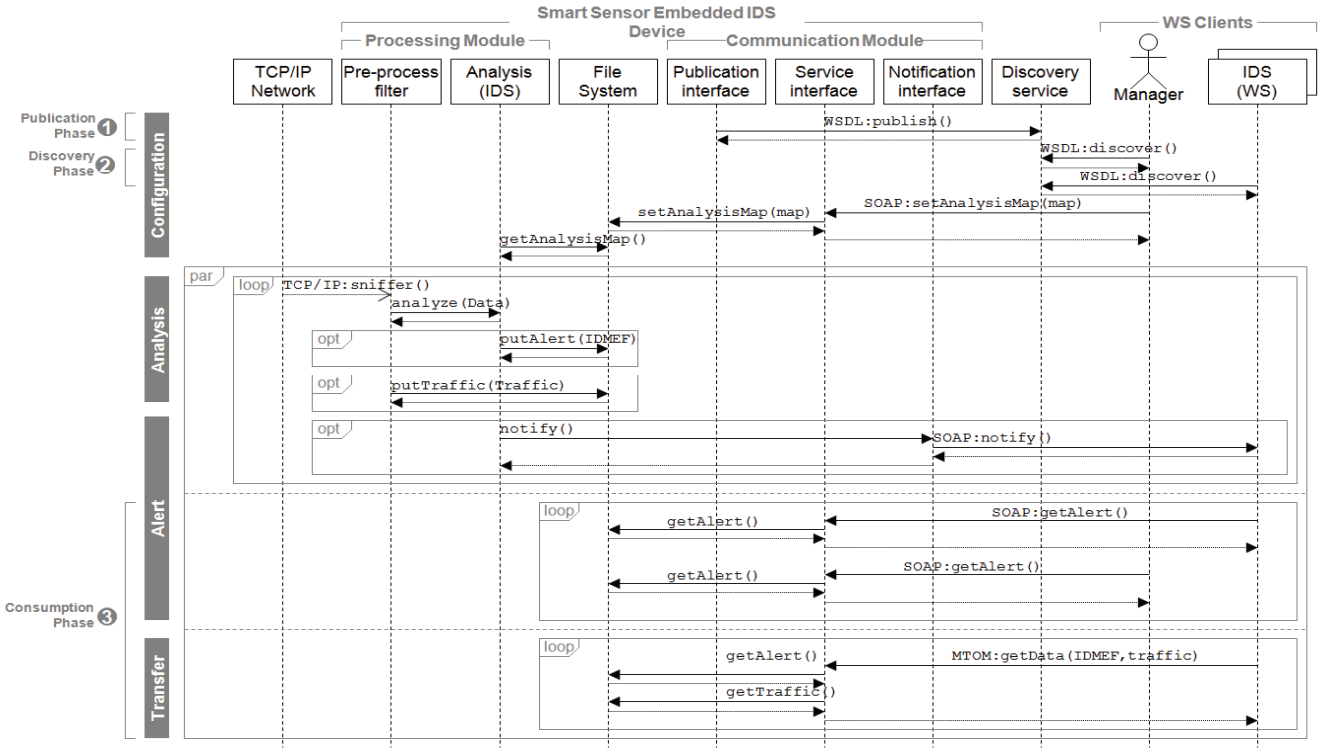


Fig. 8. Sequence diagram with the main transactions carried out during the performance test for the validation of the proposal.

*Broker*, *Service Consumer* and *Service Provider* areas.

- In the *Service Broker Area*, for the Discovery Service, we have used a computer with the Linux operating platform Ubuntu v8.04 server version with the Java-based UDDI server jUDDI v0.9rc4. For its deployment we have also installed the Tomcat Web container v5.5.23 and MySQL database v5.
- In the *Service Consumer Area*, to implement managers and clients compatible with WS, a computer based on Linux Ubuntu has been used and we have implemented a Java-based client application that allows us to discover and consume the services offered by *Smart Sensors*. The version of the JDK used is 1.6.0_02. To implement the Web services we have used Apache Axis v1.4. platform with the Tomcat Web container. The development was performed using the Eclipse Europe Release v3.3 environment along with the Web Tools module installed. To store alerts and other configuration data has also been used a MySQL database v5.
- In the *Service Provider Area* we have placed our previously described *Smart Sensor* prototype.

The working network refers to the network which we wish to monitor for possible intrusion attempts. In this network we have joined our network sensors along with the necessary equipment to re-create the different conditions of traffic and load under which we wanted to test the proposal's validity. Likewise, it has been incorporated into this working network a sample of the malicious traffic extracted from the DARPA data set to take a battery of real and objective data against which the results will be judged. This network consists of the following elements:

- A set of ten hosts with different hardware and operating platforms. These hosts will use the network with different load levels through the scheduled use of standards network services (Web, FTP, SMTP, etc.) aiming to assess the performance of the detection system based on network load.
- The embedded IDS prototypical device that acts as *Smart Sensor*, establishing its NIC in promiscuous mode to analyze all network traffic. To perform this task, the switch used to interconnect the network has been configured so that all network traffic is forwarded to the port to which the *Smart Sensor* is connected. Besides, in addition to the alerts generated and for validation purposes, we also monitored other performance variables such as: processed packets and memory and CPU usage.
- Finally, we have connected a network traffic injector that supplies DARPA network traffic used in the tests and which contains both malicious traffic as well as normal one. The application used is *tcpreplay* v3.3.1 that allows us to forward network traffic stored in files with the *pcap* format used in DARPA files.

It is important to remember that this work is not aiming to an improvement of the intrusion detection results that may have been presented in other investigations. Our goal is merely to prove that a compact, low-consumption and zero-maintenance solution can behave in a solvent manner under different system load conditions. For this reason, in the following paragraphs we discuss the tests conducted in this regard: self-management capabilities (publication, discovery, etc.) and analyzer performance under different load conditions.

### B. Discovering and publication testing

A jUDDI server has been used to check the auto-registration module. It registers the service in a standard way. Connecting the prototype to a network, it is possible to prove, by means of traffic sniffing, that it seeks the IDS service in the jUDDI server. If the service is not found then it is published, by doing the authentication to deal with jUDDI server private functions.

The service has been successfully tested using, on the one hand, a PHP client using a NuSOAP library and, on the other, a command line standard client developed by Apache group called WSIF. This standard client is able to use services through an invoker that only writes the function, the parameters and the address where WSDL sheet is stored, builds a correct call to the prototype service and shows the obtained response.

In the sequence diagram of Fig. 8, within the configuration block, we can see the major transactions involved in the test of publication, discovery and consumption:

1) Initially the *Smart Sensor* is published in the Discovery service using the UDDI protocol by Publication interface (Publication Phase).
2) Then both the Manager and an external IDS based on WS can discover the location and functionality of the *Smart Sensor* conducting an UDDI discovery process with the Discovery Service (Discovery Phase).
3) Once the sensor is discovered, it is possible to perform sensor´s management tasks from the Manager. The diagram shows the analysis map updating which will be used by the sensor Analyzer. With this aim the Manager interacts with the Service interface which stores the map in the File System, where it can be retrieved by the Analysis module (IDS).

### C. Bulk transfer and analysis engine testing

In the sequence diagram of Fig. 8, within the analysis block, we have collected the main actions involved in the analysis and consumption test:

1) In the analysis block, the Pre-process filter module gets packets that travel over the network and pre-process them. At this stage, we initially reconstruct the TCP/IP sessions analyzing the IP packets that conform it to calculate the input data of the neural network. Once obtained, and after being normalized, the data are passed to the Analysis module. In the event of an attack being detected in the analysis module, we store the relevant alert in the File System. Alternatively, if the network sensor was set to do so, the traffic associated with the alert also is stored in the File System. Along the same line, if the asynchronous notification service was asked for, it would be carried out under the previously established parameters. .
2) In the alerts block, both the Manager and the external IDS may demand to obtain the produced alerts. With this aim, the external elements get in touch with the sensor through the Service Interface responsible for recovering the alerts from the File System in order to deliver them to the client.

### D. IDS Performance testing

With the goal of evaluating our proposal, the experimental results of the application of the SOM network to the Web traffic attack detection are shown in this section. An anomaly-based service-oriented NIDS has been developed. The neural network has been trained with HTTP traffic and is able to recognize attacks against this service.

In order to obtain results that can be judged against other IDS proposals, we need to use a standardized set of proving data. To date, DARPA intrusion detection evaluation data is the most comprehensive set known to be generated for IDS assessment purposes [34]. It has been regarded within the scientific community as a significant breakthrough for the independent and scientific appraisal of any given IDS performance. Therefore, the data used for the training and testing of the implemented neural networks have been taken from the DARPA dataset.

The election of the SOM neural network topology has been carried out by calculating the two eigenvectors of the autocorrelation matrix of the training data that have the greatest eigenvalues; the relationship between the dimensions of the network is obtained considering the relationship between these two eigenvalues and the number of patterns in the training data [15]. Following these criteria, a SOM of 18 x 28 dimensions have been selected, with linear initialization within the rank of the training patterns and with a Gaussian neighborhood function.

For the networks validation after the training phase, we have used the same data again, and we compute the distance to the winning neuron for each pattern. The networks are validated if at least 95% of the input patterns vectors have a distance smaller than two standard deviations with respect to the vectors of the winning neurons [15]; this heuristic assumes a Gaussian distribution.

In order to make the results directly comparable, we have followed the author's recommendations for the networks parameters values [35]. The chosen basic parameters were: an initial learning rate of 0.9, a high initial neighborhood for better sorting and a number of learning iterations no less than 500 times the number of neurons.

Likewise, to carry out the performance comparisons, we have selected three tests corresponding to three network's load levels: low load, with only traffic injected by *tcpreplay*, which injects normal and anomalous traffic; medium load, with the previous traffic and the traffic sent by the other network nodes (1MB/s of network traffic); and high load, with injected and generated traffic by the other network nodes trying to maximize network traffic (in a network of 100 Mb/s, say 12 MB/s).

Fig. 9 shows the use of the ARM CPU. For clarity purposes, the chart reflects only 100 representative seconds from the three weeks total tests duration time. According to these tests, the average CPU usage in low load mode was about 3.9%, in medium load there was an average CPU usage of 20.5%, but in the case of high network load, the processor was saturated and the average reached a 79.9%. The graphics also show processing peaks due to specific network traffic increases. We

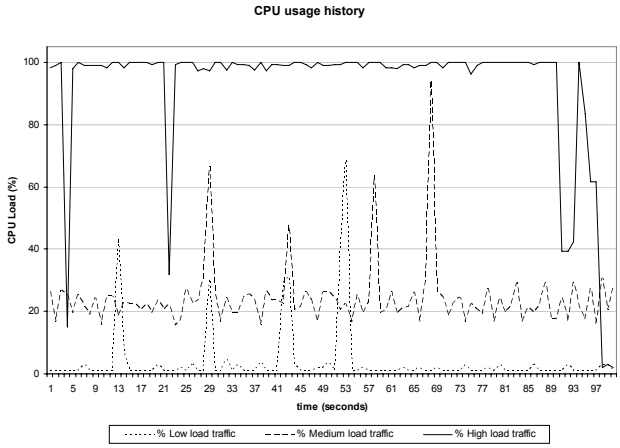


Fig. 9. Graphical representation of the CPU usage.

can conclude that this behavior is quite stable and proportional to the network load.

Fig. 10 shows the network traffic processed by the network interface of the network sensor. We have also taken 100 representative seconds from the total duration time of the tests. The average network interface load is 0.1% for low network load, 9.9% for medium network load and 68.7% for high network load. There is a clear relationship between this and the previous graphic, since the peaks in the transmission of information mirror the saturation points of CPU usage. As a wrap up it is safe to say that the network interface behavior is in direct proportion to the network load.

The graphic chart of Fig. 11 shows the system performance and detection capabilities as a function of the network load. In this case, conversely to figures 9 and 10, twelve different tests corresponding to twelve different load levels evenly allocated in a range from 1 to 12 MBps has been conducted. The *CPU usage* rises steadily until reaching a saturation point at which, while most of the time the average fluctuates around 80%, except for the occasional falls caused by network interface saturation, the rest of the time the processor is close to 100%. The processed frames curve indicates the percentage of packets processed by the application and the packets received by the network interface. The curve moves downwards as the load is increased because the network interface buffer

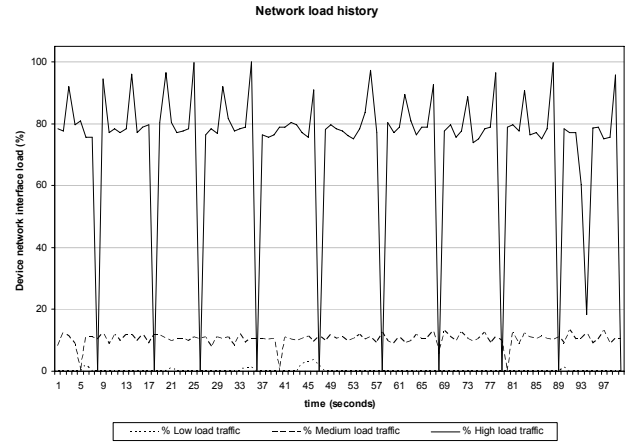


Fig. 10. Graphical representation of the device network interface load.

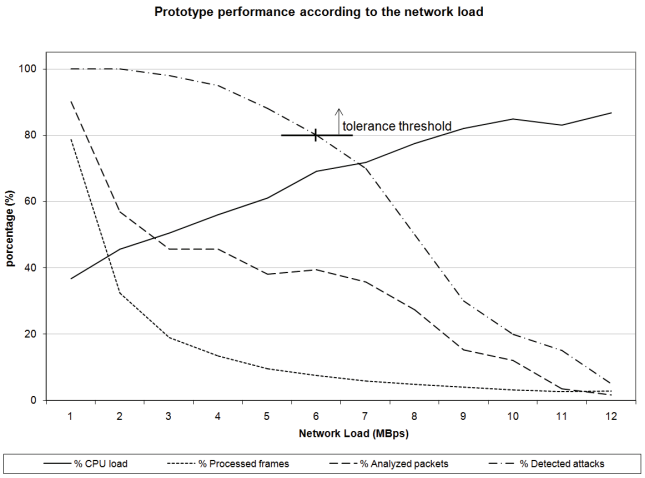


Fig. 11. Graphical representation of the prototype performance as a function of the network load.

saturation outgrows the application ability to process the information, leading to discard packets. The *analyzed packets* curve indicates the percentage of packages analyzed by the IDS regarding the injector *tcpreplay* sent packets. This curve also decreases as the application load goes up.

Finally, the performance and capacities of IDS are judged against two different ratios: false positives and detected attacks. Detected attacks curve indicates the percentage of attacks detected by the IDS. As can be seen, if we consider tolerable a detection rate up to 80% for an IDS [11], as shown in Fig. 11, the device is effective until 6MBps of network load (50%) is reached. In the field of network services is customary to consider that only below 30% of network load a networking device can guarantee its services. In addition to these results, we would like to point out that the tested IDS has generated no false positives whatsoever. In conclusion, the prototype designed in this paper outperforms by far the commonly accepted standard values for this kind of services.

As an additional note, we would like to add that the use of volatile memory has remained constant during the whole process, at a level of occupation of about 57%.

Regarding the repercussion of the use of web services in the system, as well as network traffic transmission via MTOM, different tests have been made resulting on a mean IDS traffic overload of 1300 bytes for each coded alarm notification in SOAP and a CPU pinpointed increase (1 sec. sample rate) of 2.5% per alarm. In relationship with the traffic download associated to the alarm, it is safe to say that it results highly dependent on the volume of traffic associated to each attack. Our test results have shown a mean transfer of 38.9 KB per alarm, as well as a mean of 19 alarms per hour. It should be noticed that this traffic is transferred via a second network interface available in the prototype.

## VII. CONCLUSION

This work core proposal is the design of an IDS embedded in a small network device that acts as a smart network sensor. The sensor is considered intelligent not only because of its

ability to transfer the network traffic analyzed to other DIDIS components, but also for its capacity of doing that at any moment on demand as well.

From the functional point of view, the device functionality can be considered a NIDS, generating alerts in IDMEF format and offering its functionality as Web Service. So, the proposal is in perfect harmony with the SOA model and the device is able to register itself in an external registry.

A full functional prototype has also been developed. This prototype has been used to validate the proposal. The results show that the device exhibits a very stable behavior and is capable to provide a service, as critical as the intrusion detection service is, under really adverse conditions of network traffic load.

We are currently working in a management model based on the incorporation of semantic to the definition of the services, so that high levels of automation can be reached throughout the whole configuration and management tasks process, not only for an isolated device, but also for a DIDS as well, even though the conditions, resources or necessities vary over time. In this case the basic WS specification has more limitations, so we are working on a WS-* specification which incorporates more sophisticated functionalities.

## REFERENCES


[1] X. Qin and W. Lee, "Statistical causality analysis of infosec alert data," in *Proc. Int. Sym. Recent Advances in Intrusion Detection*, Pittsburgh, PA, USA, 2003, pp. 73–93.

[2] E. E. Stelzer and T. A. Gonsalves, "Embedding RMON in large LAN switches," *IEEE Network*, vol. 13, no. 1, pp. 63–72, Jan. 1999.

[3] J. Hill, R. Szewcyzk, A. Woo, S. Hollar, D. Culler and K. Pister, "System architecture directions for networked sensors," *Operating Systems Review*, vol. 35, no. 11, pp. 93–104, Nov. 2000.

[4] J. Belenguer and C. T. Calafate, "A low-cost embedded IDS to monitor and prevent Man-in-the-Middle attacks on wired LAN environments," in *Proc. Int. Conf. on Emerging Security Information, Systems, and Technologies*, Valencia, Spain, 2007, pp. 122–127.

[5] T. Sato and M. Fukase, "Reconfigurable Hardware Implementation of Host-Based IDS," in *Proc AsiaPacific Conference on Communications*, Penang, Malaysia, 2003, pp. 849–853.

[6] K. M. Tan and R. A. Maxion, "Why 6? Defining the operational limits of stide, an anomaly-based intrusion detector," in *Proc. IEEE Sym. Security and Privacy*, Oakland, CA, USA, 2002, pp. 188–201.

[7] C. Kruegel and G. Vigna, "Anomaly detection of Web-based attacks," in *Proc ACM Conf. on Computer and Communications Security*, Washington, USA, 2003, pp. 251–261.

[8] K. Wang and S. Stolfo, "Anomalous payload-based network intrusion detection," in *Proc. Int. Sym. Recent Advances in Intrusion Detection*, French Riviera, France, 2004, pp. 203–222.

[9] S. J. Han, K. J. Kim and S. B. Cho, "Evolutionary learning program's behavior in neural networks for anomaly detection," in *Proc. Int. Conf. on Neural Information Processing*, Calcutta, India, 2004, pp. 236–241.

[10] S. Zanero and S. Savaresi, "Unsupervised learning techniques for an intrusion detection system," in *Proc. ACM Sym. Applied Computing*, Nicosia, Cyprus, 2004, pp. 412–419.

[11] F. J. Mora, F. Maciá, J. M. García and H. Ramos, "Intrusion detection system based on growing grid neural network," in *Proc. IEEE Mediterranean Electrotechnical Conference*, Malaga, Spain, 2006, pp. 839–842.

[12] R. Lippmann and R. Cunningham, "Improving intrusion detection performance using keyword selection and neural networks," *Computer Networks*, vol. 34, no. 4, pp. 597–603, Oct. 2000.

[13] J. Cannady and J. Mahaffey, "The application of artificial intelligence to misuse detection," in *Proc. Int. Sym. Recent Advances in Intrusion Detection*, Louvain-la-Neuve, Belgium, 1998, pp. 75–94.

[14] P. Lichodzijewski, A. Zincir-Heywood and M. Heywood, "Dynamic intrusion detection using self-organizing maps," in *Proc. annu. Canadian Information Technology Security Symposium*, Ottawa, Canada, 2002, pp. 93–97.

[15] M. Ramadas, S. Ostermann and B. Tjaden, "Detecting anomalous network traffic with self-organizing maps," in *Proc. Int. Sym. Recent Advances in Intrusion Detection*, Pittsburgh, PA, USA, 2003, pp. 36–54.

[16] L. Ying-Dar, T. Kuo-Kun, L. Tsern-Huei, L. Yi.Neng, H. Chen-Chou and L. Yun-Cheng, "A plataform-based SoC design and implementation of scalable automaton matching for deep packet-inspection," *Journal of Systems Architecture*, vol. 53, no. 12, pp. 937–950, Dec. 2007.

[17] C.A. Hudson, N.S. Lobo, R. Krishnan, "Sensorless Control of single switch-based switched reluctance motor drive using neural network," *IEEE Trans. on Industrial Electronics*, vol. 55, no 1, pp. 321–329, Feb. 2008.

[18] S. Jung and S. Su kim, "Hardware Implementation of a real-time nueral network controller with a DSP and a FPGA for Nonlinear Systems," *IEEE Trans. on Industrial Electronics*, vol. 54, no 1, pp.265–271, Feb. 2007.

[19] D. Zhang and L. Hui, "A stochastic-Based FPGA Controller for an Induction Motor Drive With Integrated Neural Network Algorithms," *IEEE Trans. on Industrial Electronics*, vol. 55, no. 2, pp. 551–561, Feb. 2008.

[20] C. Kruegel, F. Valeur and G. Vigna, *Intrusion Detection and Correlation: Challenges and solutions*. New York: Springer, 2005.

[21] M. E. Locasto, J. Parekh, A. Keromytis and S. Stolfo, "Towards collaborative security and P2P intrusion detection," in *Proc. IEEE Information Assurance Workshop*, West Point, NY, USA, 2005, pp. 333–339.

[22] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora and M. Miyashita, "A line in the sand: a wireless sensor network for target detection, classification and tracking," *Computer Networks*, vol. 46, no. 5, pp. 605–634, Dec. 2004.

[23] J. M. Gonzalez, V. Paxson and N. Weaver, "Shunting: a hardware/software architecture for flexible, high-performance network intrusion prevention," in *Proc. ACM Computer and Communications Security*, Alexandria, USA, 2007, pp. 139–149.

[24] V. C. Gungor and G. P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," *IEEE Trans. on Industrial Electronics*, vol. 56, no. 10, pp 4258–4265, Oct. 2009.

[25] U. Toop, P. Muller, J. Konnertz and A. Pick, "Web based Service for Embedded Devices," in *Proc. Workshop on Web, Web-Services and Database Systems*, Erfurt, Germany, 2002, pp. 141–153.

[26] B. Akin, U. Orguner, H.A. Toliyat and M. Rainer, "Phase-Sensitive detection of Motor Fault Signatures in the Presence of Noise," *IEEE Trans. on Industrial Electronics*, vol. 55, no 6, pp. 2539–2550, Jun. 2008.

[27] B. Singh, V. Verma and J. Solanki, "Neural Network-Based Selective Compensation of Current Quality Problems in Distribution System," *IEEE Trans. on Industrial Electronics*, vol. 54, no 1, pp. 53–60, Feb. 2007.

[28] S. Won, F. Golnaraghi and W. Melek, "A Fastering Tool Tracking System Using an IMU and a Position Sensor With Kalman Filters and a Fuzzy Expert System," *IEEE Trans. on Industrial Electronics*, vol. 56, no 5, pp. 1782–1792, May. 2009.

[29] K. P. Birman, S. Guha and R. Murty, "Scalable, self-organizing technology for sensor networks," in *Advances in Pervasive Computing and Networking*, B. Szymanski, B. Yener, Ed. New York: Springer, 2005, pp. 1–16.

[30] IEEE 802.3af, "(CSMA/CD) Access Method and Physical Layer Specifications Amendment: Data Terminal Equipment (DTE) Power via Media Dependent Interface (MDI)," *IEEE Computer Society*, 2003. [Online]. Available: http://www.ieee802.org. [Accessed: March 20, 2009].

[31] G. Scheible, D. Dzung, J. Endresen and J. E. Frey, "Unplugged But Connected, Design and Implementation of a Truly Wireless Real-Time Sensor/Actuator Interface," *IEEE Industrial Electronics Magazine*, vol. 1, no 2, pp. 25–34, Jul. 2007.

[32] B.K. Douglas, *Web Services and Service-Oriented Architectures: The savvy manager's guide*. San Francisco: Morgan Kaufmann, 2003.

[33] H. Debar, D. Curry and B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)," *Internet Engineering Task Force*, 2007. [Online]. Available: http://www.ietf.org. [Accessed: March 20, 2009].

[34] DARPA Intrusion Detection Evaluation. [Online]. Available: http://www.ll.mit.edu/IST/ideval/index.html. [Accessed: July 25, 2009].
[35] T. Kohonen, *Self-Organizing Maps*. Berlin: Springer, 2001.

**Francisco Maciá-Pérez** (M'08) was born in Spain in 1968. He received his engineering degree and the Ph.D. degree in Computer Science from the University of Alicante in 1994 and 2001 respectively.

He worked as System's Administrator at the University of Alicante form 1996 to 2001. He was an Associate Professor from 1997 to 2001. Since 2001, he is an Assistant Professor and currently he is the Director of the Department of Computer Science and Technology at the University of Alicante. His research interests are in the area of network management, computer networks, smart sensor networks and distributed systems, which are applied to industrial problems.

**Francisco J. Mora-Gimeno** (M'08) was born in Spain in 1967. He received the M.Sc. degree in Computer Science from the Polytechnic University of Valencia, Valencia, Spain, in 1995. He received the Ph.D. degree in Computer Science from the University of Alicante in 2010.

Since 2002, he has been an Assistant Professor with the Department of Computer Science and Technology, University of Alicante. His main topics of interest include intrusion detection systems, network security, computer networks and distributed systems.

**Diego Marcos-Jorquera** (M'08) was born in Spain in 1974. He received his engineering degree and the Ph. D. degree in Computer Science from the University of Alicante in 1999 and 2010 respectively.

He is currently an Assistant Professor with the University of Alicante. His research interests are in the area of network management, computer networks, and distributed systems.

**Juan Antonio Gil-Martínez-Abarca** was born in Spain in 1970. He received his engineering degree in Computer Science from the University of Alicante in 1994.

Since 1998, he is System's Administrator at the University of Alicante and, since 1999, he has been an Associate Professor at the Department of Computer Science and Technology at the University of Alicante. His research interests are in the area of network management, computer networks and distributed systems.

**Héctor Ramos-Morillo** was born in Alicante, Spain, in 1978. He received the engineering degree in Computer Science from the University of Alicante in 2004, where he has been working toward the Ph.D. degree in the Department of Computer Science and Technology since 2005.

He is currently a System's Administrator at the Department of Computer Science and Technology, University of Alicante. His research interests are in the area of network management, computer networks, embedded systems and smart sensor networks.

**Iren Lorenzo-Fonseca** was born in Cuba in 1982. She received her Engineering and Master degree in Computer Science from the José Antonio Echevarría Institute of Technology (CUJAE) in 2005 and 2007 respectively and her Ph. D. degree in the Department of Computer Science and Technology of the University of Alicante 2010.
She is currently Professor at the Computer Science Faculty of the José Antonio Echevarría Institute of Technology. Her research interests lay in the area of artificial intelligent, computer networks and distributed systems.