Sensornets 2014
3rd International Conference on Sensor Networks, 7-9 January, 2014, Lisbon - Portugal

Resilient Supervision System over WSAN: A Distributed Multi-Agent Architecture

Fábio Januário[1,3], Amâncio Santos[2,3], Catarina Lucena[1],
Luís Palma[1], Alberto Cardoso[3] and Paulo Gil[1,3]
[1]*Departmento de Engenharia Electrotécnica, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Portugal*
[2]*Instituto Superior de Engenharia de Coimbra (ISEC), Portugal*
[3]*Centre for Informatics and Systems (CISUC), University of Coimbra, Portugal*
f.januario@campus.fct.unl.pt, {lbp, psg}@fct.unl.pt, amancio@isec.pt, cml@uninova.pt, alberto@dei.uc.pt

Keywords: Wireless Sensor and Actuator Networks, supervision systems, outliers detection, resilient systems, multi-agent systems, distributed computing.

Abstract: Wireless Sensor and Actuator Networks enable flexibility, low operational and maintenance costs, as well as scalability in a variety of scenarios. However, in the context of remote control and monitoring applications the use of Wireless Sensor and Actuator Networks can impact the system's performance due to several factors, such as outliers in sampled raw data, communication breakdown or owing to security issues. In order to improve the overall system's resilience, this paper proposes a distributed hierarchical multi-agent architecture where each agent is responsible for a specific task. Experimental results collected from a laboratory test-bed show the relevance of incorporating the proposed methodology in the context of monitoring and networked control systems over Wireless Sensor and Actuator Networks.

1 INTRODUCTION

Wireless Sensor and Actuator Networks (WSANs) have attracted considerable attention in the last few years. They are distributed networks of sensors and actuators nodes, which act together in order to monitor and/or control a diversity of physical environments or systems (Mendes et al., 2009). Each node is a small electronic device with wireless communication capabilities, including data storage and processing power. In addition, they can still be programmed to interact with the physical environment by means of built in sensors and actuators (Opina et al., 2009).

In industrial environments WSANs can be used in supervision applications, which contribute to reducing installation and operation costs (Cerrada et al., 2007). However, constraints on resources, in particular limited processing power, small data storage, narrow bandwidth and autonomy may lead to inaccurate data being sent to the sink (Tirkawi and Fischer, 2009). Moreover, these systems suffer from two main types of vulnerabilities. The first one refers to the problem of monitoring the system for faults and failures, and responding in such a way to prevent and/or mitigate the problem. A second issue involves the vulnerability to cyber-intrusion, in which malignant actors can mask the system's degradation or provide "fake" data to higher management levels, concerning the current system's status (Rieger et al., 2012). This kind of vulnerabilities can, to some extent be mitigated by implementing dedicated "coadjutants", which contribute to make the overall system more robust and resilient.

In this work a resilient system is regarded as a control system that maintains state awareness and an acceptable level of operational normalcy in response to disturbances, including threats of an unexpected and malicious nature (Garcia et al., 2012). In the case of nodes' deployment in harsh environments, they may exhibit malfunction behaviours, which ultimately result in corrupted raw data that are subsequently sent to the sink (Zang et al., 2010). These errors in the readings are, as a whole denoted as outliers, and should be cancelled out or accommodated for the sake of the supervision systems' performance.

The implementation of resilient enforcement mechanisms can be carried out through the implementation of dedicated algorithms and heuristics, within an framework based on agents (Cardoso et al., 2012; Rieger and Villez, 2012). The incorporation of these

"coadjutants" in the context of faults and failure diagnosis, or to make the system less vulnerable, is very attractive as it makes possible to implement a cooperative and decentralized agent-based framework for such purposes, by making use of agents' inherent features, namely autonomy, reactivity, pro-activity, cooperation and intelligence.

Several agent-based methodologies and architectures have been proposed to support the design of resilient control systems (see e.g. (Rieger and Villez, 2012)). Generally, these methodologies are based on a combination of intelligent techniques, such as fuzzy systems, encryption algorithms, Bayesian networks or neural networks, just to name a few. Given the limited memory and computation power constraints of nodes, however, these schemes can hardly be implemented on WSAN based applications.

In the context of WSANs environments, the prevalent line of research has been focussed on the use of mobile agents to tackle particular problems, such as routing, clustering and localization (Chen et al., 2007). Jamont and Occelo (Jamont and Occello, 2007) propose an original method for designing embedded multi-agent system, while in Xiong and Bai (Xiong and Bai, 2010) several generalized frameworks are proposed to address some critical issues in Wireless Sensor Networks (WSNs) applications, namely interoperability, time synchronization, power management and distributed computation. Worth to mention is the work of Freitas et al. (Freitas et al., 2009), which focusses on an agent-based framework, acting as an integral part of a middleware, to support autonomous setup and adaptation of sensor networks.

Although these architectures may solve a number of known problems associated with WSANs, they do not allow the network to adapt to physical environment changes. Regarding monitoring and control applications, apart from addressing security and distributed computation issues, it is also crucial to ensure the integrity of the collected data, and to guarantee the system is operating safely in case of network communication faults.

With these premises in mind, the present work proposes a new resilient enforcement architecture. It is based on a hierarchical multi-agents paradigm, where each agent is tailored for executing specific and coordinated tasks, namely to outliers detection and accommodation, as well as to maintain the system in a safe state operation, in case of communication link breakdown, and to account for security issues under the form of manipulation of nodes' configuration by a malicious actor. The effectiveness of the proposed framework and its impact on the supervision system performance is empirically assessed through experiments conducted on a test-bed.

2 MULTI-AGENT SUPERVISION SYSTEM

Agents can be regarded as computing entities in a given environment, presenting a certain degree of autonomy, and possessing the ability to feel and act in order to accomplish a given mission. Some of their inherent features include the following (Paolucci and Sacile, 2005):

- Autonomy: agents are independent entities, able to accomplish a given task, without any programming or direct intervention;
- Reactivity: capability of perceiving their environment and respond quickly and effectively to changes;
- Pro-activity: ability to take initiative goals and behave in order to meet them;
- Cooperation: agents have the ability to interact and communicate with one another for the sake of their own teleonomy;
- Intelligence: in order to evaluate and take over a task, in an autonomous way, an agent should incorporate intelligent techniques;
- Mobility: agents have the ability to move its code from a node to another one in a system, offering mobility properties in distributed computational devices.

Taking into account these features, it is possible to devise a cooperative and decentralized multi-agent framework with a number of advantages for WSANs applications, namely, for systems' monitoring and control, while keeping the system running in a safe mode, in case of communication loss between the server and nodes, or in the presence of cyber attacks.

2.1 Architecture Overview

The proposed architecture concerns the implementation of a resilient supervision system over WSANs, for which the issues of communication, network topology and routing protocols have already been tackled.

The supervision system is composed of three main components (Fig. 1): a multiple inputs and multiple outputs (MIMO) system being monitored or controlled, a computer (server) running the middleware, the dispatcher software, which feeds data from the

WSAN into the middleware and other top-level applications, such as the monitoring application and the remote controller, along with the communication infrastructure, based on a WSAN. Each node is a processing device that collects information from the environment through attached sensors or transmitters, sends sensor reading and reports to the sink, and delivers received control actions or actuation signals, stemming from the remote controller through the WSAN, to the plant.

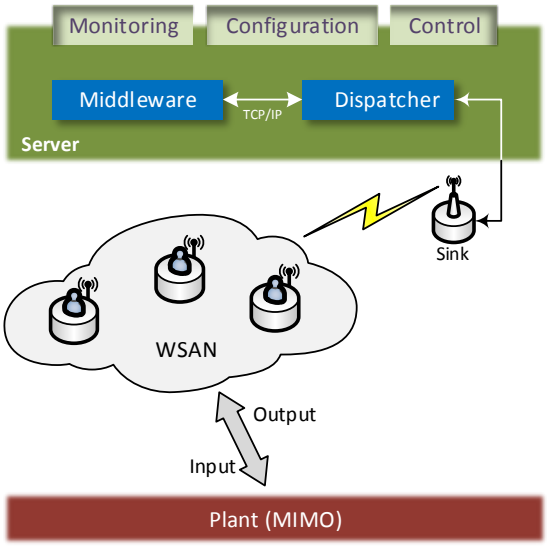

Figure 1: System Architecture.

2.2 Local Multi-Agent System

The architecture is based on the multi-agent paradigm, where each agent is responsible for a specific task. As shown in Fig. 2, each local node includes a set of agents for monitoring, safety control, as well as for management and supervision purposes. The following sections describe each local agents' main features.

2.2.1 Master Agent

The main goal for this agent is to carry out extensive management routines related to other subordinate local agents, and to coordinate the communications between the node and the sink. When this agent is activated, it automatically launches dependent lower-level agents (actuator node or sensor node agents, see Fig. 2). This agent is also responsible for monitoring the status of all local dependent agents and, in case of an agent crash, the master will relaunch the corresponding "thread".

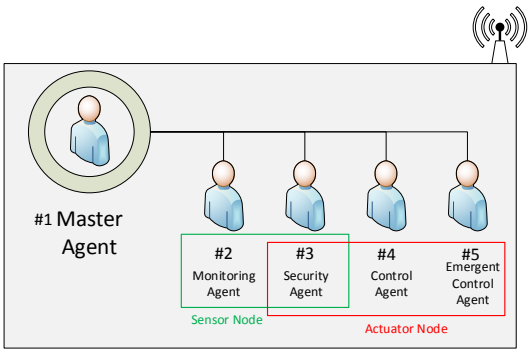

Figure 2: Local Multi-Agent System Architecture.

2.2.2 Monitoring Agent

This agent is responsible for collecting data from the environment and for accommodating possible outliers in raw data. The detection and accommodation of outliers is based on the approach proposed in (Cardoso et al., 2012).

2.2.3 Security Agent

This agent is responsible for periodically check all important variables in the system, as well as messages structure. If any anomaly is detected, an alarm is triggered and sent by the master agent to the user interface. In addition, the system is switched to safe mode operation.

2.2.4 Control Agent

The purpose of this agent is to send to the digital-to-analog converter (DAC) control actions received from the sink. This agent is also responsible for testing the periodicity of control actions, through which a communication breakdown is detected.

2.2.5 Safety Agent

The function of this agent is to maintain the system in a safe operation mode, in case of communication link breakdown, and controller's malfunction or user errors. This goal is achieved by defining a threshold for normal operation in terms of system's outputs. When the safety agent detects that collected data are outside the pre-specified bounds or no control actions have been received from the sink in the corresponding sampling interval, the system is switch to safe mode operation, and a local control action computed based on an on-off heuristic, while considering the most recent reference signal received from the middleware.

2.3 Multi-Agent framework

The multi-agent framework relies on a collaborative and sharing profile/approach, which is a necessary condition to any distributed system. Each node incorporates a set of agents (see Fig. 2), including a master agent responsible for communicating with the user interface and with local agents, or even with other WSAN nodes.

Each message comprises a header and a payload. The header is composed of the sender and destination addresses in the WSAN, message sequence number, hops, and a control identifier. The message payload (see Table 1) contains the *Message Type*: the message can stem from the system's application or from a local agent; *Node ID*: denoting the node address; *Control ID*: the command flag for local agents; *Data ID*: data collected in the node ID; *Agent ID:* agent's identifier that will be launched, stopped or resumed; *Agent MSG:* data provided by an agent.

Table 1: Messages Payload.

Type	Node ID	Control ID	Data ID	Agent ID	Agent MSG

2.3.1 Master Agent framework

Table 2 shows the messages' parameters received by the master agent and sent by sink. When launching a master agent, it is required the sampling time T_s and the sigma factor for outliers detection in monitoring agents (σf). In order to start an actuator agent, it is necessary to send the sampling time T_s, the maximum level Υ_u and minimum level Υ_o to be used by the safety agent. These values, excluding the sample time, can be changed later on by sending new "Upload Limits" and "Upload Sigma Factor", through dedicated messages.

The security agent parameters are defined internally in the master agent during the programming node, and are not accessible by the user, for the sake of the system's security.

Table 2: Messages received by the Master Agent.

Sensor			
Start Agent		T_s	σf
Upload Sigma Factor		σf	-
Actuator			
Start Agent	T_s	Υ_u	Υ_o
Upload Limits	Υ_u	Υ_o	-
Control Action	k	$u(k)$	$r(k)$

In order to implement the control action on the plant, the sink sends a message to each actuator nodes, comprising the input voltage, along with the current reference for each of the tanks under control, namely T_1 and T_2. These references will be used by the safety control agent in case of communication breakdown.

When the master agent receives a "Start Agent" #ID message, the corresponding local agent is launched, which sends periodically messages to the master agent reporting its operation state. If the master agent does not receive periodically reports from a running agent this agent is assumed to be "off" and launched once again.

The messages sent by the master agent to the user application interface via the sink node are presented in Table 3. They include the security state $\psi(k)$ of the node and the accommodated reading $y(k)$ in the case of a sensor node, or the safe control action $u_e(k)$ in case of an actuator node.

Table 3: Messages sent by the Master Agent.

Sensor			
Send State	k	$y(k)$	$\psi(k)$
Actuator			
Send State	k	$u_e(k)$	$\psi(k)$

2.3.2 Monitoring Agent framework

Monitoring agents collect the level of each tank through the analog-to-digital converter (ADC) port to which the level sensor is connected, and based on an outlier detection and accommodation algorithm, proposed in (Cardoso et al., 2012) each sample is accordingly tested (see Algorithm 1). If a sample is tested positive, it is replaced by the mean value of the oversampled time series, excluding those samples that are outside the threshold limits (σf providing a way to control threshold size).

2.3.3 Security Agent framework

The security agent is responsible for checking the values of T_s, σf, Υ_u, Υ_o, $y(k)$, $u(k)$ and $u_e(k)$. This agent has internally defined maximum and minimum values that these variables can take. Whenever they are outside the admissible threshold an alarm message is triggered and sent to the user interface. In addition, the variable is adjusted to be within the pre-defined limits. The structure of the messages sent and received by the node and between agents is quite stringent. If the security agent detects a different structure a message error is triggered and sent to the user inter-

Algorithm 1: Monitoring Agent.

Input: fs − Sampling Frequency
fos − Oversampling Frequency
σf − Sigma Factor

$Ts \leftarrow 1 / fs$; // sampling period
$Tos \leftarrow 1 / fos$; // oversampling period
while *true* **do**
  $ti \leftarrow$ `GetTime`; // initial time
  $i \leftarrow 0$; // initial iteration
  repeat // oversampling until next sample time
    $i \leftarrow i+1$; // iteration
    $z[i] \leftarrow$ `GetSample`; // oversample
    `Sleep` (Tos) // node sleep for Tos seconds
  until `GetTime` $\geq ti + Tos$;
  $y \leftarrow$ `GetSample`; // sample
  $z[i+1] \leftarrow$ `Mean`(z); // expected value
  $\Delta u \leftarrow z[i+1] + \sigma f \times$ `STD`(z); // upper threshold
  $\Delta l \leftarrow z[i+1] - \sigma f \times$ `STD`(z); // lower threshold
  if $y \notin [\Delta l, \Delta u]$ **then** `SendToMaster`(y); // the sample is not an outlier
  else `SendToMaster`$(z[i+1])$; // outlier: accommodated
end

face, while the current message will be ignored (see Table 4).

Table 4: Security Agent Send Messages

Variable	Error code ($\psi(k)$)
T_s	1
σf	2
Υ_u	3
Υ_o	4
$y(k)$	5
$u(k)$	6
$u_e(k)$	7
Message structure	8

2.3.4 Control Agent framework

The control agent is started by the master agent, from which receives as configuration parameters the sampling time and the initial control action. This agent uses the sampling time to test the network for communication breakdown. If in a given sampling interval the agent does not receive any control action the master agent is "warned" and the safety agent will take over the control of the system (see Algorithm 2).

Algorithm 2: Control Agent.

Input: Ts − Timeout
u − Initial Control Action

`SetADC` (u) ; // dispatch to ADC the initial control action
$Time.timeout \leftarrow Ts$; // define timeout
$Time.init \leftarrow 0$; // define initial time
$Time$.start; // launch the Time counter
while *true* **do**
  $ev \leftarrow$ `Event`; // wait for new event
  if $ev = Timeout$ **then** // timeout event
    Enabled Safety Agent;
  end
  if $ev = NewControlAction$ **then** // control action received event
    $u \leftarrow$ `GetControlAction`;
    `SetADC` (u); // dispatch to ADC the control action
    $Time.reset$; // reset the counter
  end
  `Sleep` (Ts)
end

2.3.5 Safety Agent framework

This agent receives the level of the tank provided by the corresponding sensor node and implements a relay controller (on/off). Furthermore, if the liquid level in the controlled tanks exceeds the prescribed bounds this agent also takes over the control system (see Algorithm 3).

3 CASE STUDY

3.1 Test-bed Description

The test-bed consists of a three-tank system (Fig. 3), a remote desktop computer, where the user interface and controller is running, six Crossbow TelosB nodes (five for tanks and one sink). The main goal is to monitor and control the liquid level in the tanks T_1 and T_2, while maintaining the system under safety operation mode, in case of faults events.

The AMIRA© DTS 200 benchmark three-tank system (Fig. 3) comprises three plexiglas cylindrical tanks with identical cross-section supplied with distilled water. Liquid levels, namely h_1, h_2 and h_3 are measured by piezoresistive transducers. The middle tank T_3 is connected to the other two tanks by means

Algorithm 3: Safety Agent.

Input: Υ_u − Maximum Level
Υ_o − Minimum Level
Ts − Waiting time
$flag$ − Flag that indicates communication breakdown

```
1  Δ_t ;                                   // level threshold
2  Δ_u ;                                   // control action increment
3  while true do
4      y ← GetLevel;                       // get level
       u ← GetControlAction;               // get last control action
       r ← GetRefrence;                    // get last reference
5      if y > Υ_u − Δ_t then SetADC (u − Δ_u);
       // decrease pump
6      else if y < Υ_o + Δ_t then SetADC (u + Δ_u);
       // increase pump
7      else if flag = 1 then               // communication breakdown
           if y > r + Δ_t then SetADC (u − Δ_u);
           // decrease pump
           if y < r − Δ_t then SetADC (u + Δ_u);
           // increase pump
       end
8      else disable emergent control;      // normal state
   end
```

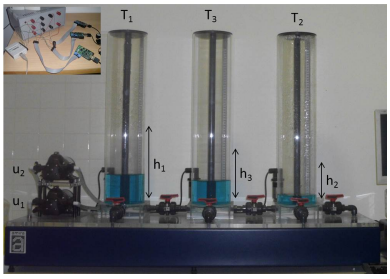

Figure 3: Three-tank system.

of circular cross section pipes provided with manually adjustable ball valves. In the tank T_2 it is located the main outlet of the system, which is directly connected to the collecting reservoir by means of a circular cross-section pipe provided with an outflow ball valve. This system has two pumps to circulate water from the bottom reservoir to tanks T_1 and T_2.

In what the WSAN is concerned, it is built in with Crossbow TelosB (TPR2400) sensor nodes, which are ultra-low power wireless devices. These nodes leverage industry standards like USB and IEEE 802.15.4 to interoperate seamlessly with other devices. Each TelosB node has available analogue and digital ports, to which sensors and actuators are attached. The operating system used in WSAN programming is based on de Contiki OS. This operating system has been written in C language with support for dynamic loading and replacement of individual programs and services. Additionally, it was built around an event-driven kernel, but provides optional preemptive multithreading, which can be applied to individual processes (Dunkels et al., 2004).

In the case of the present setup, three nodes are configured as sensors, in order to collect tanks' levels (h_1 h_2 and h_3), while two other nodes are used as actuators, associated with pumps P_1 and P_2. In addition a sixth node is used as a sink/gateway. This node is attached via USB port to the remote desktop computer, thus allowing its communication with the WSAN nodes. As can be seen in Fig. 4 these nodes are hierarchically arranged in a N-5 topology, but allowing peer-to-peer communication between sensor and actuator nodes associated with tank T_1 and T_2.

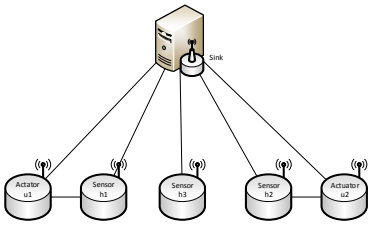

Figure 4: Logical topology

Finally, the remote controllers were implemented based on a PID-Fuzzy controller (see (Januário et al., 2013)), being the control actions and readings delivered through the WSAN.

3.2 Experimental Results

This section is devoted to assess the proposed methodology in dealing with vulnerabilities on the supervision systems over WSANs, as mentioned in Section 1.

3.2.1 Outliers accommodation

The first experiment was designed to assess the benefits of outliers accommodation for the supervision system resilience. In this context, readings have been taken from sensor nodes' ADCs, at the frequency of 1 Hz, and using also a USB data acquisition board in parallel, namely a NI6008, from National Instruments, through which samples have been taken at the same discrete times. A root mean square of error (RMSE) metric (1) was subsequently compute to

compare three different σf against the raw data. Table 5 shows the corresponding results.

$$RMSE = \sqrt{\frac{\sum_{k=1}^{N} [y_n(k) - y_b(k)]^T [y_n(k) - y_b(k)]}{\sum_{k=1}^{N} y_b(k)^T y_b(k)}} \quad (1)$$

with y_n the sensor reading sent through the WSAN and y_b the reading sampled via the USB data acquisition board.

As can be observed in Table 5, the RMSE measure drops with the decrease of the underlying sigma factor, which is in line with what was expected as the thresholds depends on the sigma factors.

Table 5: Performance metric for the tree-tank system.

	σf - null	$\sigma f - 2$	$\sigma f - 1.5$	$\sigma f - 1$
Tank 1	0.0241	0.0237	0.0223	0.0182
Tank 2	0.0398	0.0396	0.0386	0.0379
Tank 3	0.0321	0.0310	0.0302	0.0297

3.2.2 Safety agent operation

Figure 5 and Figure 6 show the liquid levels in tanks T_1 and T_2 when a safety control agent is in operation. In this case it was configured to a minimum level of 0.15 % and a maximum level of 0.45 %. As can be observed, the local safety agent does not allow the liquid level outside the $[0.15, 0.45]$ % even if the reference signal for the control system is set outside. Inside the admissible system's outputs the safety agent is in idle operation mode.

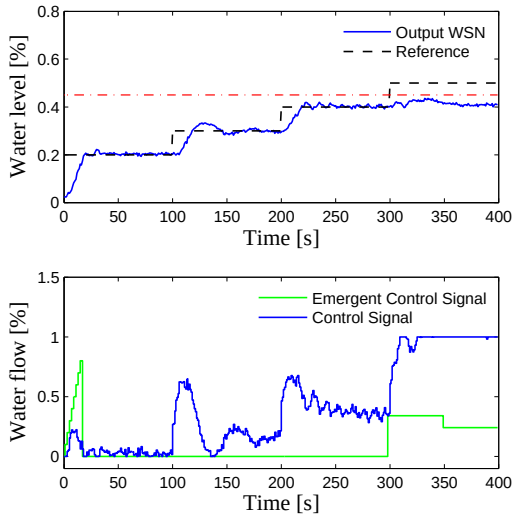

Figure 5: MIMO system working out of limits Tank 1.

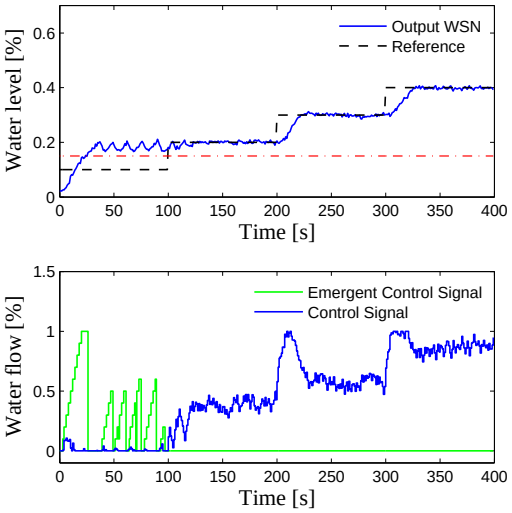

Figure 6: MIMO system working out of limits Tank 2.

In the event of a communication link breakdown, from the sink to the actuator node, the safety agent is able to deal transparently with this situation, by maintaining the tank level in the vicinity of the last reference received. Figure 7 refers to the case of communication loss (fail zone) between 150 and 250 second and from 350 to 450 second. As can be observed, the safety agent maintains the tank level around the last reference signal received from the sink node by means of an on/off control approach, and without exceeding the prescribed admissible level limits.

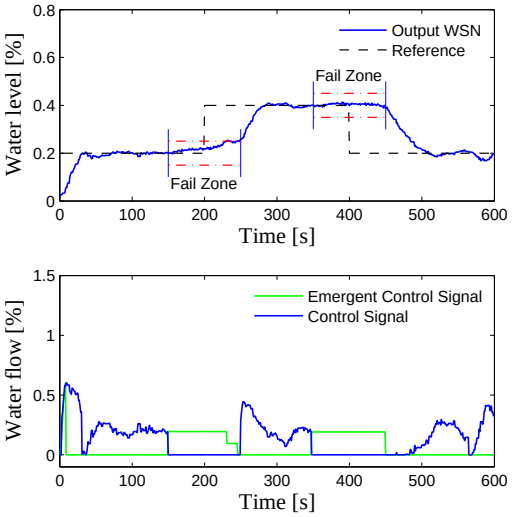

Figure 7: Communication faults: Tank 1.

4 CONCLUSIONS

This paper addressed the problem of supervision over Wireless Sensor and Actuator Networks from the resilience point of view, namely, focusing on outliers detection and accommodation, safety operation and breakdown communication links. The proposed approach was based on a hierarchical multi-agent system framework deployed in wireless nodes. Outliers are detected using statistical based techniques, along with an oversampling technique prior to periodic sensor reading, while the Wireless Sensor and Actuator Networks security is achieved by periodically "scanning" the control variables and messages structure. Breakdown communication faults and controller malfunction have also been considered. Results from experiments on a test-bed comprising a benchmark three-tank system and a WSAN built with Crossbow TelosB nodes reveal the feasibility and relevance of the proposed framework. Finally, it should be mentioned that this work is still under progress. Further developments will include security policies against malicious attacks and other features, such as self awareness in heterogeneous environments.

ACKNOWLEDGEMENTS

Januário, F. acknowledge Fundação para a Ciência e Tecnologia (FCT), Portugal for the Ph.D. Grant SFRH/BD/85586/2012. This work has been partially supported by iCIS-Intelligent Computing in the Internet of Services, Project CENTRO-07-ST24-FEDER-002003.

REFERENCES

Cardoso, A., Santos, A., Nunes, G. B., and Gil, P. (2012). A multi-agent approach for outlier accommodation in wireless sensor and actuator networks. *Controlo2012 - 10th Portuguese Conference on Automatic Control*.

Cerrada, M., Cardilho, J., Aguilar, J., and Faneite, R. (2007). Agents-based design for fault management systems in industrial processes. In *Computers in Industry*.

Chen, M., Gonzalez, S., and Leung, V. C. M. (2007). Applications and design issues for mobile agents in wireless sensor networks. *Wireless Communications*, 14:20–26.

Dunkels, A., Grönvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors*, Tampa, Florida, USA.

Freitas, E. P., Heimfarth, T., Ferreira, A. M., Wagner, F. R., Pereira, C. E., and Larsson, T. (2009). An agent framework to support sensor networks setup and adaptation. *Computer Science and Information Technology*, pages 619–626.

Garcia, H. E., Lin, W., and Meerkov, S. M. (2012). A resilient condition assessment monitoring system. In *5th International Symposium on Resilient Control Systems (ISRCS)*.

Jamont, J. P. and Occello, M. (2007). Designing embedded collective systems: The diamond multiagent method. *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, 2:91–94.

Januário, F., Santos, A., Lucena, C., Palma, L., Cardoso, A., and Gil, P. (June 2013). Outliers accommodation in fuzzy control systems over wsan. In *5th International Conference on Intelligent Decision Technologies (KES-IDT), Portugal*.

Mendes, M. J. G. C., Santos, B. M. S., and Costa, J. S. (2009). Multi-agent platform and toolbox for fault tolerant networked control systems. *Journal of Computers*, pages 303–310.

Opina, A. P., Canola, A. M., and Carranza, D. O. (2009). Integration model of mobile intelligent agents within wireless sensor networks. In *IEEE Latin-American Conference on Communications*.

Paolucci, M. and Sacile, R. (2005). *Agent-based Manufacturing and Control System: New Agile Manufacturing Solutions for Achieving Peak Performance*. CRC Press.

Rieger, C., Zhu, Q., and Basar, T. (2012). Agent-based cyber control strategy design for resilient control systems: Concepts, architecture and methodologies. In *5th International Symposium on Resilient Control Systems (ISRCS)*, pages 40–47.

Rieger, C. G. and Villez, K. (2012). Resilient control system execution agent (recosea). In *5th International Symposium on Resilient Control Systems (ISRCS)*.

Tirkawi, F. and Fischer, S. (2009). Generality challenges and approaches in wsns. In *I. J. Communications, Networks and System Sciences*.

Xiong, F. Y. and Bai, L. (2010). Interoperable wireless sensor network model using multi-agent-based middleware. *International Symposium on Intelligent Signal Processing and Communication Systems,*, pages 1–4.

Zang, Y., Meratnia, N., and Havinga, P. (2010). Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communnications Surveys & Tutorials*, pages 159–170.