# Distributed Multi-Agent Architecture for Dependable Supervision over WSAN: Oil Refinery Tests

Paulo Gil and Luis Palma
UNINOVA-CTS, Dep. Eng. Electrotécnica, FCT/UNL
Monte de Caparica, Portugal
{psg,lbp}@fct.unl.pt

Amâncio Santos
Instituto Politécnico de Coimbra
ISEC, DEIS
Coimbra, Portugal
amancio@isec.pt

Alberto Cardoso
CISUC-Informatics Engineering Department
University of Coimbra
Coimbra, Portugal
alberto@dei.uc.pt

## ABSTRACT

In supervision applications over wireless sensor and actuator networks the communication channels' reliability, along with the network quality of service and readings are critical issues. As such, in the design and implementation stages of such schemes preventive measures should be taken into account for the sake of control systems dependability. This paper proposes a multi-agent based framework implemented in some of the nodes of the network, for dealing with intermittent communication link breakdowns on the forward channel and accommodating outliers on the raw data taken from the environment. Experiments on a real oil refinery process demonstrate the feasibility of such approaches in the context of critical industrial supervision tasks.

## Categories and Subject Descriptors

B.4.5 [**Reliability, Testing, and Fault-Tolerance**]: Built-in tests; B.8 [**Performance and Reliability**].

## General Terms

Algorithms, Performance, Experimentation

## Keywords

Oil refinery plant, wireless sensor networks, outliers and communication faults, dependability, multi-agent systems

## 1. INTRODUCTION

In the current technological context automation and control tasks are of paramount importance to industry. Commonly, they rely on wired communication infrastructures, for sending sensor readings to a remote control room, where Supervisory Control and Data Acquisition (SCADA) systems are used to monitor and manipulate a given plant or process. The approach based on wired solutions, however, is recognised to be rather awkward in what the flexibility, deployment and configuration are concerned. In fact, the installation is quite costly in terms of equipment, materials and labour. On the other hand, in the case of critical plants, the security measures overhead adds-up, as cables must be laid out underground or embedded into buildings' structures. As a result of such a rigidity it is not easy to make reconfigurations to the original system in order, for instance, to adapt to new layouts or processes. To overcome these limitations associated with wired based communication solutions, researchers and practitioners, alike have shifted to wireless based architectures to connect embedded devices, such as sensors, actuators and control platforms. One of such an approach relies on Wireless Sensor and Actuator Networks (WSANs).

WSANs are based on small, low-cost sensor or actuator nodes, spatially distributed over a given environment, where each node consists of a miniature electronic device with wireless communication interface, power source, programmable micro controller, and possibly multi-type sensors or ADC and/or DAC ports. The small dimension of these devices, absence of wired connections and external power supply, easy deployment, flexible installation and fully mobile operation, make them ideal for a wide range of applications, particularly in the process industry (see e.g. [8]).

Sensor and actuator nodes may be arranged in single-hop or multi-hop networks with a variable number of self-organized devices. Sensor nodes can be considered in monitoring applications, through which data is gathered from a given target, while actuator nodes are incorporated in networked control systems or in automation contexts. Although WSANs are conceptually feasible from the academic point of view, many concurrently challenges still prevent their dissemination on the industrial theatre. Among these obstacles may be reckoned the robustness to cyber-attacks, confidentiality of transmitted data, performance assurance in high packet delivery rates, reliability of active links, or the quality of raw data [4].

As for sensor nodes, due of harsh environmental conditions, along with constraints on nodes' resources it is likely to observe artefacts on the raw data [3], which are commonly denoted as outliers. These outliers should be accommodated in order to avoid biasing decisions. Most of the available techniques for detecting these artefacts are computationally intensive, requiring large amounts of memory, high level of energy consumption, noticeable communication overheads, and above all are not tailored to heavy online data streaming. Regarding actuator nodes integrated on centralised networked control system, it might happen that because of channels link breakdown the closed loop system behaviour


*MoMM* '14 December 08-10-2014, Kaohsiung, Taiwan

http://dx.doi.org/10.1145/2684103.2684110

end up being inevitably compromised.
Dealing with outliers in raw data and intermittent communication faults on forward channels are the main motivations and contributions of this work. Both issues are addressed by appealing to a hierarchical multi-agent system deployed on nodes. Outliers detection and accommodation is implemented using univariate statistics, along with an oversampling technique, while fault tolerant mechanisms are incorporated by applying a given command action, in such a way the system is driven to a safe operation state.
These approaches aiming at strengthening the dependability of the overall system, namely in terms of purity/quality of readings and tolerance to intermittent communication faults is evaluated on a real oil refinery process.

# 2. MULTI-AGENTS SYSTEMS

## 2.1 Agents and Multi-Agents

An intelligent agent can be defined as a computing entity embedded in a given environment, comprising a certain degree of autonomy and having perception abilities. They possess inherent capabilities to interact with congenerous agents, while presenting idiosyncratic pro-activity features, in the sense that it can assume initiatives in a way to persistently fulfil its own goals. Therefore, an agent is supposed to act or react spontaneously, executing pre-emptive and independent actions, in accordance to some predefined goals [5].
Although in some particular cases agents can act on their own, it is common to find clusters of agents, in different functions, carrying out a number of tasks. In this sense, a multi-agent system (MAS) is a collection of heterogeneous intelligent agents organised in a particular topology, with cooperation and interaction capabilities, aiming to accomplish, in a holistic manner, a given prescribed mission.
The MAS architecture implemented in this work follows a hierarchical multi-agent system (HMAS) topology. Fig. 1 shows a local HMAS topology for monitoring and control over WSANs, in which several agents responsible for monitoring and control, using available digital-to-analogue converters (DACs) ports, are locally coordinated by its master-agent. The master-agent's main purpose is to carry out extensive management routines related to lower-level agents and to monitor the communication status between local nodes, and the WSAN's sink or gateway. Agents belonging to the lowest layer are used to interface directly, and in a distributed way, with sensor readings or to take over the command of the system in case of link breakdown. As such, they process queries, pre-process collected data from the environment, extract useful information from these data, as instructed by upper-level agents, which have the responsibility for coordinating subordinate entities, based on a centralised goal-driven conditioning strategy.

## 2.2 Multi-Agent Architecture

The hierarchical multi-agent based architecture is composed of two layers, namely a higher-level layer with coordination functionalities and a bottom layer comprising subordinate agents that are committed to specific tasks,such as monitoring analogue-to-digital converters (ADCs) readings, detection of outliers and their subsequent accommodation or switching the system to a safe operation regime. Available lower-level agents can be independently launched, paused, resumed and stopped at runtime. Additionally, the agents' configuration can be remotely performed by means of appropriate command messages delivered through the sink node.

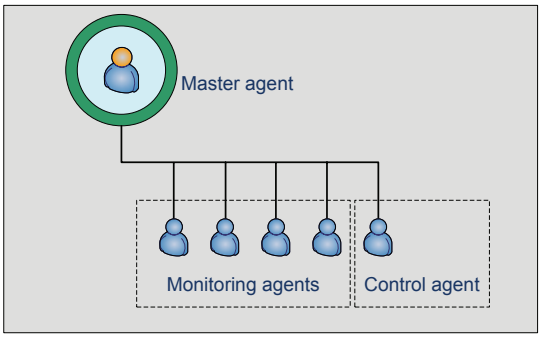


**Figure 1: Illustration of a multi-agent system.**

**Table 1: Message Payload.**

| Message type | Node ID | Control ID | Data ID | Agent ID | Agent msg |
|---|---|---|---|---|---|

Each message comprises a header and a payload. The header includes the sender and destination addresses in the WSAN, message sequence number, hops, and a control identifier. Regarding the message payload (see Table 1) it consists of *Message Type*: the message can be originated 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.
Concerning the agents' commands, the platform is provided with: *i*) *Start Agent*: starts a given agent at by sending the command flag, the agent's ID and the node's destination address; *ii*) *Stop Agent*: stops a particular agent; *iii*) *Start All Agents*: starts all agents stacked at node's memory; *iv*) *Stop All Agents*: stops all the agents that are running at the sensor or actuator node.

### 2.2.1 Master-Agent

Its main commitments are related to coordination tasks, namely managing routines associated with lower-level agents: launching, stopping and resuming agents, and their configuration; monitoring the communication channel between the underlying node and sink, using a two-way beaconing system; sending alarms to the base-station. By default, this agent is always launched whenever the sensor node is switched on or rebooted. When started, it runs a number of diagnostic routines, and launches predefined subordinate agents, aiming at real-time monitoring of raw data, outliers detection and accommodation, as well as the control agent responsible for handling communication faults. Complementary, the master-agent is also committed in monitoring the "health" of lower-level agents. In case of one of the agents crashes the master-agent will kill the corresponding thread, reloads the agent's "clone" from the local repository, and eventually starts it. In addition, if a given agent is no longer needed, the master-agent removes it from memory, by killing the corresponding thread.

### 2.2.2 Monitoring Agents

Subordinate monitoring agents are launched by the master-

**Algorithm 1:** Control agent.

**Input**: $us$ − Safety Command Value
$ts$ − Safety Time
$fs$ − Sync Flag

```
StartTimer(t);                                   // timer init
while true do                                    // infinite cicle
    c ← GetMessage;                              // get new command
    if c = abort then return;                    // abort
    if c ≥ 0 then                                // DAC command
        Dac(c);                                  // send to DAC
        if (t > ts) & fs then                    // DAC was in safety mode
            SendMessage(t);                      // send sync message
        end
        ResetTimer(t);                           // reset timer
    else if t > ts then                          // timer expired
        Dac(us);                                 // put DAC in safety mode
    end
end
```

**Algorithm 2:** Outliers detection and accommodation.

**Input**: $fs$ − Sampling Frequency
$fos$ − Oversampling Frequency
$snd$ − Standard Normal Deviate
**Output**: Sample/Accommodated Sample

```
Ts ← 1 / fs;                                     // sampling period
Tos ← 1 / fos;                                   // oversampling period
ti ← GetTime;                                    // initial time
i ← 0;                                           // initial iteration
repeat                                           // oversampling until next sample time
    i ← i + 1;                                   // iteration
    z[i] ← GetSample;                            // oversample
    Sleep(Tos);                                  // node sleep for Tos seconds
until GetTime ≥ ti + Ts;
y ← GetSample;                                   // sample
z[i + 1] ← Mean(z) ;                             // expected value
Δu ← z[i + 1] + snd × STD(z);                    // upper threshold
Δl ← z[i + 1] − snd × STD(z);                    // lower threshold
if y ∉ [Δl, Δu] then return y;                   // the sample is not an outlier
else return z[i + 1];                            // is an outlier: accommodated
```

agent. They are started with a number of parameters as argument in order to detect and accommodate possible outliers in the readings, by using the univariate statistical approach presented in Section 3. These parameters include the sampling and oversampling frequencies, along with the standard normal deviate. If the sample taken at a given sampling time $(k+1)\,T_s$ is tagged as an outlier, the corresponding value is replaced by the mean of the oversampled data, and an alarm is accordingly triggered.

### 2.2.3 Control Agents

The control agent is loaded by the master agent, as well, from which it receives at launch-time, as configuration parameters, safety time ($t_S$), the control action corresponding to a safe mode operation ($u_S$) and also a flag. In a time-out event the control agent will take over the system's manipulation by sending it to a safe operation state, according to Algorithm 1.

## 3. OUTLIERS ACCOMMODATION

As the problem to be addressed here concerns the local real-time detection and accommodation of outliers in readings, the algorithm must be computationally efficient, which dictated the choice for a statistical based approach.

The univariate statistical-based approach to limit sensing assumes that a single physical variable is observed through a sensor reading, and a given time-series used to determine boundary thresholds for in-control operation [1]. The violation of these limits by sampled data will indicate a likely presence of an outlier. The upper ($\Delta_u$) and lower ($\Delta_l$) limits on the Shewhart chart are particularly critical to the performance of the method, namely in terms of sensitivity and specificity. More specifically, tight threshold limits will result in a high rate of false outliers, while loose ones increase the missed detection rate.

Statistical hypothesis theory can be used to predict false outlier and missed detection rates on a given *ensemble*. Let $z$ be a monitoring variable, for which any deviation from its mean, $\bar{z}$, results from additive errors, and assume that its variability follows a Gaussian distribution $\mathcal{N}(\bar{z}, \sigma^2)$ with standard deviation $\sigma$. Then the probability P of $z$ lying within a given interval is expressed as

$$\mathrm{P}\left\{z < \left(\bar{z} - c_{\alpha/2}\sigma\right)\right\} = \mathrm{P}\left\{z > \left(\bar{z} + c_{\alpha/2}\sigma\right)\right\} = \frac{\alpha}{2} \qquad (1)$$

$$\mathrm{P}\left\{\left(\bar{z} - c_{\alpha/2}\sigma\right) \leq z \leq \left(\bar{z} + c_{\alpha/2}\sigma\right)\right\} = 1 - \alpha \qquad (2)$$

where $c_{\alpha/2}$ is the standard normal deviate corresponding to the $(1 - \frac{\alpha}{2})$ percentile, and $\alpha$ the level of significance, which specifies the degree of trade-off between false outlier and missed detection rates. Some typical values for the standard normal deviate include $c_{\alpha/2} = \{1.0; 1.5; 3.0\}$. In the case of a standard normal deviate $c_{\alpha/2} = 1.5$, the probability $\mathrm{P}\left\{\left(\bar{z} - c_{\alpha/2}\sigma\right) \leq z \leq \left(\bar{z} + c_{\alpha/2}\sigma\right)\right\}$ is 86.64 %.

In order to improve the consistency of the univariate statistical approach, the present work resorts to an oversampling technique within each sampling time, namely for $t \in [kT_s, (k+1)\,T_s]$, with $T_s$ the sampling period, $t$ the continuous time and $k$ the discrete-time. This enables collecting a statistically representative time series, which is subsequently used within the underlying Shewhart control chart. If the sample taken at time $(k+1)\,T_s$ lies outside the threshold limits it is tagged as an outlier and accordingly replaced by a plausible estimate. This methodology is outlined in Algorithm 2.

## 4. REFINERY CASE-STUDY

The proposed approach to deal with outliers detection and accommodation in WSANs was implemented in an oil refinery plant, namely at the Petrogal refinery, located at Sines, Portugal. The refinery environment is highly challenging with respect to wireless communications, as huge thick metal structures and non-stop operating machines contribute to a high noise level, which can seriously hamper the WSAN performance and the quality of raw data. Experiments on site have shown that the radio environment, although noisy, was fairly stable.

Another issue concerning the WSAN's deployment on this

**Figure 2: Water treatment area - Line 4.**

particular scenario is its need to comply with strict rules for the access and management of personnel in critical environments. Several areas are classified as ATmosphère EXplosive (ATEX), which imposes access and mobility restrictions to personnel, and require that all electrical equipment, including sensor nodes and other hardware, must be encased in ATEX-certified boxes.

## 4.1 Scenario Description

The test-bed included a WSAN consisting of a number of nodes deployed at the water treatment area, Line 4 (Fig. 2) and a sink located at the portable office, which is attached to a Linux operating system computer. In this machine a Dispatcher software processes packets received from the WSAN, and forwards each packet via a TCP/IP connection to the GINSENG middleware running on an another computer. This computer is located at the refinery main control room. Additionally, the Dispatcher relays configuration commands, originated from the middleware to individual sensor/actuator nodes, being the middleware responsible for the configuration of the whole network and local multi-agent architecture (Section 2.2).

The WSAN consisted of Crossbow's TelosB (TPR2400) nodes, namely twelve sensor/actuator nodes and one sink. These devices are IEEE 802.15.4 (ZigBee) compliant, and incorporate eight 12-bit ADC ports, two 12-bit ADC ports associated with the internal thermistor and battery voltage and two 12-bit DAC port. In addition, they include two light transducers and one humidity/temperature transducer on-board connected to 14-bit ADC ports. The nodes were enclosed in ATEX junction boxes attached to external 9 dB antennas, in order to cope with the wireless signal attenuation using the on-board (internal) TelosB nodes' antenna. Fig. 3 shows one of the deployed nodes at the water treatment area within an ATEX junction box (JB). The junction boxes associated with the corresponding nodes are listed in Table 2.

Concerning the operating system for WSANs programming, it was developed under GINSENG project [6] and is based on the Contiki OS. It is built around an event-driven kernel, although providing optional pre-emptive multi-threading functionalities, which can be applied to individual processes [2]. Flow and pressure transmitters are connected to a $4-20$ mA current loop backbone, which extends across the water treatment area. Before providing their signals to nodes ADCs ports it is needed to convert the current-based signal to $0-2.5$ V, using dedicated current-to-voltage converters. On the other hand, to ensure a reliable and deterministic readings delivery, the communication in the WSAN was implemented based on a novel Time Division Multiple Access (TDMA) scheme [7], with respect to the Media Access Control (MAC) protocol, while the network was arranged on a multi-hop hierarchical tree routing topology for channel access. It assumes a carefully planned node deployment within a virtual tree topology. Hierarchical addresses are used to identify each node's position in a 3-3 tree topology, as depicted in Fig. 4. The nodes are programmed satisfying a 3-3 network envelope, where the sink has 3 children nodes, with each of these nodes also having 3 children nodes. As such, the nodes N-1-0, N-2-0 and N-3-0 of the first level are responsible for forwarding messages originated at the associated set of 3 children nodes, of the second level, to the sink, and configuration messages emanated by the middleware through the sink to lower level nodes, as well as control actions to actuator nodes. This schedule proved to support end-to-end network reliability of around 99 % [4].

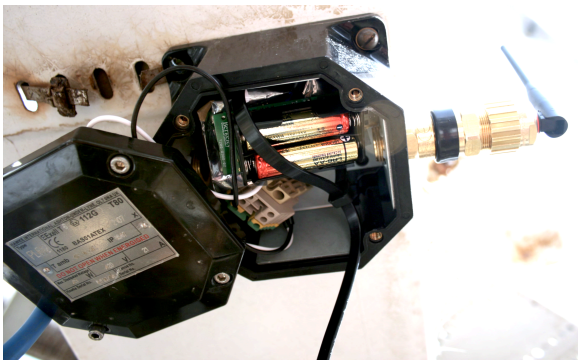

**Figure 3: Sensor node in a ATEX junction box.**

**Table 2: Junction boxes deployed in the water treatment zone.**

| JB | Type | Transmitter | Node Id | Tree |
|---|---|---|---|---|
| JB1 | - | (sink) | 101 | N-0-0 |
| JB7 | Flow | FT-5147 | 107 | N-1-0 |
| JB5 | Flow | FT-5141 | 102 | N-1-1 |
| JB12 | Pressure | PT-5170 | 108 | N-1-2 |
| JB10 | Flow | FT-5147 | 110 | N-1-3 |
| JB8 | Pressure | PT-5170 | 109 | N-2-0 |
| JB15 | Pressure | PT-5100 | 112 | N-2-1 |
| JB14 | Pressure | PT-5100 | 105 | N-2-2 |
| JB16 | Flow | FT-5100 | 113 | N-2-3 |
| JB3 | Flow | FT-5130 | 111 | N-3-0 |
| JB4 | Flow | FT-5141 | 106 | N-3-1 |
| JB13 | Flow | FT-5130 | 103 | N-3-2 |
| JB6 | Flow | FT-5100 | 104 | N-3-3 |

## 4.2 Experiments

The first experiment was aimed at comparing the consistency of readings associated with sensor nodes ADC against those provided by the standard wired solution. It was carried

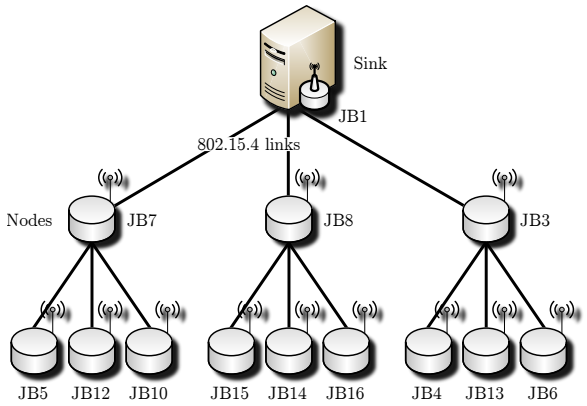


**Figure 4: Logical Topology.**

**Table 3: Comparative accuracy statistics.**

| Statistics | Variable | Value | Rel. Value |
|---|---|---|---|
| Count | WSN | 1200 | 100 % |
| Count | Backbone | 1200 | 100 % |
| Max. | \|Error\| | $4.1\times10^{-1}$ *bar* | $3.4\times10^{0}$ % FSR |
| Min. | \|Error\| | $1.5\times10^{-4}$ *bar* | $1.3\times10^{-3}$ % FSR |
| Mean | \|Error\| | $6.1\times10^{-2}$ *bar* | $5.0\times10^{-1}$ % FSR |
| $\sigma^2$ | \|Error\| | $1.7\times10^{-3}$ *bar* | $1.4\times10^{-2}$ % FSR |
| Corr. | | 0.999 | |

**Table 4: Message payload.**

| $k$ | $y_k$ | $\sigma^2$ | $\Delta_u$ | $\Delta_l$ | alarm | $z_k$ |
|---|---|---|---|---|---|---|

out using the signal from the transmitter PT-5170, which is attached to the ADC0 of the sensor node N-1-2 (ID#109). As can be observed in Fig. 5, it is clear a slight mismatch between the wired delivered PT-5170 signal and the node ID#109 readings that were sent to the sink. The signal degradation can partly be explained by the presence of artefacts in the raw data delivered through the WSAN, which is corroborated by the metrics presented in Table 3, where the error is defined as the difference between both time-synchronised signals.

The next experiment concerns the application of the proposed approach to deal with artefacts on readings taken from a pressure transmitter attached to a ADC port. To allow the detection and accommodation of outliers in raw data the corresponding monitoring agent was launched by the master-agent, at start-up. The readings were taken every 1 s ($T_s = 1$ s), while the oversampling frequency $f_{os}$ was considered to be of 100 Hz and the standard normal deviate chosen as 1.5.

The message payload associated with the sensor readings that is transmitted to the sink (see Table 4) includes: the sample number ($k$); reading ($y_k$); oversampling variance ($\sigma^2$); upper threshold for the oversampling data ($\Delta u$); lower threshold ($\Delta l$); triggered alarm (0 for undetected outlier, 1 for a sample violating of the upper threshold, and 2 for a violation of the lower threshold); the accommodated sample ($z_k$), in case of an outlier alarm is raised.

Experimental results taken from the pressure transmitter

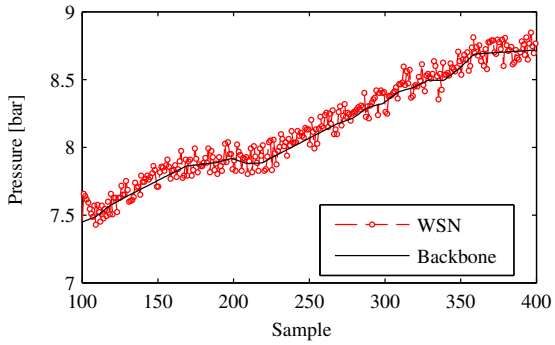


**Figure 5: Comparison between the wired PT-5170 signal in JB08 and sensor node (ID#109).**

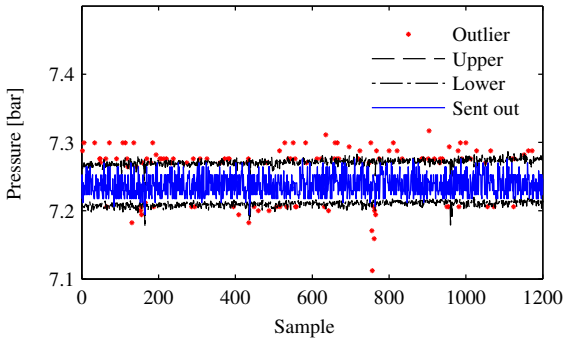


**Figure 6: Detection and accommodation of outliers in the refinery flow transmitter PT-5100.**

PT-5100, located at JB15 are shown in Fig. 6, where detected outliers are represented by a red asterisk, while the green line concerns the data sent to the sink. After being identified, outliers are accordingly replaced by the mean of oversampled data (see Algorithm 2). As can be observed, the implementation of the proposed solution towards data cleaning proves quite effective in improving the quality of the data. This is corroborated by the metrics presented in Table 4.2.

**Table 5: Pressure transmitter PT-5100 (JB15).**

| Test | Variables | Abs. Value | Rel. Value |
|---|---|---|---|
| Count | samples | 1200 | 100% |
| Count | outliers | 456 | 12.7% |
| Minimum | accommod. | $-0.13$ *bar* | $-1.1$% FSR |
| Maximum | accommod. | 0.08 *bar* | 0.6% FSR |
| Mean | accommod. | 0.01 *bar* | 0.1% FSR |
| $\sigma$ | accommod. | 0.04 *bar* | 0.3% FSR |

The final experiment concerns the evaluation of the suggested approach to deal with intermittent forward communication channel breakdown in the context of a networked control systems. As describe in Section-2.2, whenever the control agent detects a link breakdown, which occurs right after the time counter exceeds a user prescribed time-out value, this agent takes over the process control in such a way that the system is sent to a safe operation mode. It

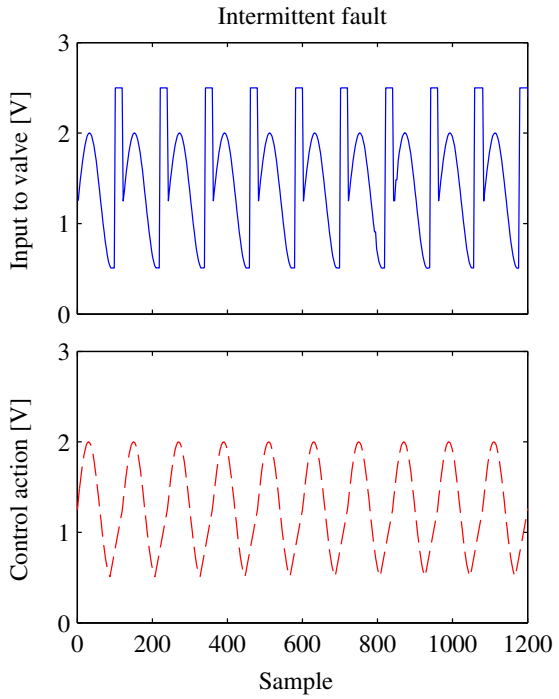


**Figure 7: Communication fault accommodation.**

should be mentioned that a predefined control action is associated to a given safe operation state. This will guarantee a dependable functioning, according to the *a priori* defined safe state, as long as the fault is active, while assuming no disturbances will be presented. For the refinery tests a motorised valve was chosen in one of the branches of the water treatment area. This valve was fed with a discrete-time sinusoidal signal with amplitude 0.75 V and off-set 1.25 V, delivered through the WSAN to node N-3-0 (ID#111). In this case the sampling rate was chosen as 2 s.
Fig. 7 shows the control signal, for which some faults were injected in the downlink communication. The injected faults were selected as an intermittent broken link sequence on the forward channel. Whenever a fault is detected by the underlying control agent it reacts accordingly by delivering a high-level output signal, $u_S = 2.5$ V, (safety measure) to avoid the controlled system becomes unpredictable.

## 5. CONCLUSION

This paper focussed on real-time data cleaning and dependability issues in wireless sensor and actuator networks. Both problems were dealt with by means of a distributed multi-agent framework, deployed to sensor and actuator nodes. Each agent was programmed to carry out specific and concise functions, namely for monitoring readings or to implement particular control policies. In the case of monitoring agents, the detection and accommodation of outliers in raw data was implemented by making use of univariate statistics along with Shewhart control charts. In the case of fault tolerance to intermittent forward channel breakdown, which is implemented by the control agent, it relies on detecting a time-out event with respect to received control actions from the remote controller. When a breakdown in the forward link is detected, a sequence of actions are triggered, through which the system is sent to a predefined safe operation mode. In order to assess the feasibility of proposed methodologies, a number of tests were conducted on an oil refinery process, specifically on a water treatment system. Results from these experiments proved both the practicability and relevance of the proposed approaches.

## Acknowledgment

This work has been partially supported by the European Commission under the contract FP7-ICT-224282 (GINSENG) and Project CENTRO-07-ST24-FEDER-002003 (iCIS-Intelligent Computing in the Internet of Services).

## 6. REFERENCES

[1] L. Chiang, E. Russell, and R. Braatz. *Fault Detection and Diagnosis in Industrial Systems*. Heidelberg: Springer, 2001.

[2] A. Dunkels, B. Grönvall, and T. Voigt. 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, November 2004.

[3] V. C. Gungor and G. P. Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE Transactions on*, 56(10):4258–4265, 2009.

[4] T. O'DONOVAN, J. Brown, F. Buesching, A. Cardoso, J. Cecilio, J. DO Ó, P. FURTADO, P. GIL, A. JUGEL, and W.-B. PÖTTNER. The ginseng system for wireless monitoring and control: Design and deployment experiences. *ACM Transactions on Sensor Networks*, 10(3), 2014.

[5] E. Oliveira, K. Fischer, and O. Stepankova. Multi-agent systems: which research for which applications. *Robotics and Autonomous Systems*, 27(1):91–106, 1999.

[6] W.-B. Pottner, L. Wolf, J. Cecilio, P. Furtado, R. Silva, J. Silva, A. Santos, P. Gil, A. Cardoso, Z. Zinonos, J. do O, B. McCarthy, J. Brown, U. Roedig, T. O'Donovan, C. Sreenan, Z. He, T. Voigt, and A. Jugel. Wsn evaluation in industrial environments first results and lessons learned. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–8, June 2011.

[7] P. Suriyachai, J. Brown, and U. Roedig. Time-critical data delivery in wireless sensor networks. In R. Rajaraman, T. Moscibroda, A. Dunkels, and A. Scaglione, editors, *Distributed Computing in Sensor Systems*, volume 6131 of *Lecture Notes in Computer Science*, pages 216–229. Springer Berlin/Heidelberg, 2010.

[8] F. Tirkawi and S. Fischer. Generality challenges and approaches in wsns. *I. J. Communications, Networks*

*and System Sciences*, 1:1–89, 2009.