WSNs Data and Configuration Management in Sensor Clouds with Cloud File Synchronization Services

Lucas Iacono[1,2,4], Carlos García Garino[1,4], Osvaldo Marianetti[3,4], and Cristina Párraga[3]

[1] *ITIC, Universidad Nacional de Cuyo, Mendoza, Argentina*
{liacono@uncu.edu.ar, cgarcia@itu.uncu.edu.ar}
[2] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina*
[3] *Facultad de Ingeniería, Universidad de Mendoza, Mendoza, Argentina*
{osvaldo.marianetti, cristina.parraga}@um.edu.ar
[4] *Facultad de Ingeniería, Universidad Nacional de Cuyo, Mendoza, Argentina*

Abstract

Sensor Clouds have opened new possibilities for researchers of disciplines such as environmental monitoring, precision agriculture and flood prevention. This technology uses Wireless Sensor Networks (WSNs) to collect real world data and Cloud Computing to store and process them. The remote management of WSNs setup and data in Sensor Clouds implies: real time access to collected data, sensor setup reconfiguration and sensor battery status monitoring. Currently there are different platforms for WSNs data and setup management in Sensor Clouds. Generally, these platforms require that scientists of the mentioned disciplines must have knowledge of WSNs and web services programming in order to reconfigure the sensors setup. Hence, these sensing resources can not be provided in a transparent way to end-users. In this paper, we propose the use of standard Cloud File Synchronization Services (CFSS) for carrying out the full management of WSNs in Sensor Clouds. In order to validate our proposal, we conduct experiments using a Sensor Cloud platform based on CFSS called Sensor Cirrus.

Keywords: Cloud Computing, WSN, Cloud File Synchronization Services, IoT, Sensor Cloud Management

1   Introduction

In lately 90s a military technology called Wireless Sensor Networks (WSNs) goes to academics field thanks to a research program called *"Berkeley Wireless Research Center PicoRadio Project"* [1]. This program consists in the development of wireless networks composed by tiny embedded systems called sensor nodes. These devices are composed by a microcontroller, memory, sensors, battery and a radio.

WSNs are targeted to work in outdoors conditions, therefore battery-life in sensor nodes must be maximized. Communication protocols for WSNs are specifically designed in order to reduce energy consumption in sensor nodes. Due to WSNs protocols are not compatible with TCP/IP stack, it is necessary to include a device called base station which acts as a gateway between WSNs protocols and TCP/IP.

Among others, the main advantages of WSNs are free maintenance, coverage of large areas for monitoring applications and programming "over the air". These particular features of WSNs have opened new possibilities for researchers which need the collection of real world data in wide areas. Currently WSNs are applied to environmental monitoring [2], precision agriculture [3], healthcare [4], flood prevention [5] and others.

Generally, these applications generate large volumes of data. As an example, can be mentioned an environmental monitoring application like [6], where a sensor node can generate up to 8 GB of data each day. Other applications generate less data volumes per sensor node (such as the ones for agricultural and home monitoring). However, the number of sensor nodes of these applications increases each day, generating large data volumes.

In order to store and process the data of the mentioned applications, it is necessary the use of high performance computing (HPC) resources like mainframes, clusters, Grid or Cloud Computing. The main advantage of Cloud Computing compared with other HPC technologies is the scalability for the provision of high performance computing resources.

The use of Cloud for storing and processing data collected by WSNs has generated large-scale infrastructures called Sensor Clouds [7]. Currently, remote WSNs data and setup management in Sensor Clouds is performed by cloud services especially designed for this function. Among others web services, SOAP-XML [8], REST-JSON [9], and commercial cloud services [10] customized for WSN remote management, can be used.

The WSNs data and setup management with

these ad-hoc cloud services have two main drawbacks. The first one is the implementation of these services for people (meteorologists, agronomists and environmental scientists) without experience in sensor networks and web services programming. In SOAP-XML and REST-JSON, users need to know how to program web services and describe data in specific formats such as XML or JSON. The second is the incorporation of WSNs already deployed and operating to Cloud Computing. This is because the addition of new software components (like the ones of XML and JSON) to these WSNs could generate fails and other performance problems.

In the last five years the appearance and popularity of cloud file synchronization services (CFSS) like Google Drive [11] and Dropbox [12] have opened new possibilities for the remote management of WSNs in Sensor Clouds. Considering that WSNs resources can be expressed as files, they can be exported to Cloud Computing using CFSS in a easy and reliable way. In CFSS users only need access to a folder provided by the CFSS, in which WSN data are stored. Besides, WSN data are stored in the same format in which they are generated by the sensors (e.g XBee API frame for Digi XBee sensors). Hence, there is no need to express them in specifics formats such as XML or JSON because standard frame parsers provided by sensor vendors can be used.

In this work we discuss and validate the implementation of CFSS for WSN remote management in sensor clouds. In order to validate the CFSS performance, we carry out an experiment using a Sensor Cloud platform called Sensor Cirrus [19] and an experimental WSN.

This paper is organized as follows. Section 2, discusses previous work on remote management of WSNs in Sensor Clouds. Next, Section 3 details Sensor Cirrus and the experimental WSN. In Section 4 an experiment to study the packet loss rate in the communication process of Sensor Cirrus is carried out. This experiment allows to validate the efficiency of CFSS for WSN management. Finally, Section 5 details the results of the conducted experiment and Section 6 brings the conclusions of this work.

2   Previous Works

Different authors have proposed solutions for the management of WSNs in Sensor Clouds. Most of the works reviewed in this section use SOAP - XML or REST-JSON web services for solving WSNs data and setup management in Sensor Clouds. Otherwise, CFSS are still a novel approach for WSN data and setup management, thus CFSS are currently less used than the other

ones.

2.1   SOAP - XML

In Tangible Cloud Computing [13] the authors applied the main concepts of Cloud (virtualized resources, SaaS and pay-per-use price model) to create a platform for integrating devices with sensing capabilities to Cloud Computing. Tangible Cloud Computing exports WSN data to Cloud Computing using Amazon web services. In addition, this platform process sensor nodes data [14] with Amazon EC2 instances, which are provided on demand according application processing requirements. WSNs data are exported to Cloud Computing using Amazon web services.

Other authors like Hori et al. [10] use commercial Cloud Services to store and process WSNs data. The used services solve the WSNs resources management. In adittion, the platform developed by Hori et al. allows the integration with other Cloud Services like the ones for business management, production history, traceability and good agricultural practice systems.

Aneka [15], is a Cloud Computing platform that allows the management of WSNs data in Sensor Clouds. This platform uses hybrid Cloud resources to process data gathered by sensors and other embedded devices. These data are delivered to Aneka by different providers in a XML, JSON or CSV format.

Ahmed and Gregory [8], develop an integration framework between WSNs and Cloud Computing using web services. The authors suggest that Sensor Clouds allow the storage of WSNs data in publics domains, which implies an efficient data usage policy.

In this subsection different platforms which use XML for WSN data and setup management have been reviewed. Authors like [13, 10], use XML services on sensor nodes. While this allows interoperability practically since data are acquired on-field, the size of sensors data increases due to XML format. Hence, radios must be on transmit state more time when sensor nodes sent XML data, increasing energy consumption. Other authors [15, 8] avoid extra energy consumption on sensor nodes using XML only on WSN base station. However, this can be a several problem in base stations based on embedded systems powered with batteries. In summary, XML services are suitable for WSN management if they are embedded only in WSN base stations connected to the power grid.

2.2   REST - JSON

Xively [16] is a commercial platform for Internet of Things, which expands the concept of "WSN share

data in public Clouds". Xively includes storage and visualization of WSNs data for the interaction of companies with their employees, products and customers along all the production process. This platform uses REST services implemented in the Xively REST API.

In [17], Sheng et al. Develop a lightweight Framework for IoT device management. This framework has two main components. The first one is a CoAP (constrained Application Protocol) component over 6lowPAN sensor nodes for communication and device management. The second one is a multiprotocol CoAP - REST gateway, embedded in 6lowPAN border gateways. This multiprotocol gateway allows WSN data and configuration management from Cloud Computing. Because, the REST service is embedded only in the gateway of the WSN, the integration to Cloud is performed conserving CoAP and 6lowPAN low resources consumption on sensor nodes.

Hirafuji et al. [9], implement an Ambient Sensor Cloud System for agricultural applications. This platform uses Twitter Cloud services for WSNs data storage and access. The main goal of this platform is the simple and low cost concept proof of Twitter usage for solving the management of big data collected by large WSNs.

In conclusion, REST-JSON services allow interoperability and low energy consumption. This is because data formated according JSON are less verbose than the XML ones. Hence, the transmission of JSON data implies less energy consumption in sensor nodes than XML data.

2.3 CFSS

In [18] the authors implement a platform based on CFSS for accessing to environmental monitoring WSNs using Dropbox. This platform uses Dropbox for storaging WSNs data on Cloud Computing, but not for sensor nodes setup management.

The main advantages of CFSS are reliability, data ubiquity and easy data share. First, CFSS brings reliability because provides data back-up on both: Cloud and users devices. Second, synchronized data are ubiquous, they can be accessed by several users from different places through a wide range of devices. Finally, CFSS allows easy data share. WSN providers put both data: sensors measurement and configuration parameters on folders, which are synchronyzed with the Cloud and clients get data from same folders. The main problem of some popular CFSS clients like Dropbox and Google Drive is the lack of compatibility whith low-cost embedded systems like Arduino (which are widely used as base stations in WSN). This issue can be solved using a remote machine (external to WSN) in which CFSS is running and receiving data from WSN base station through
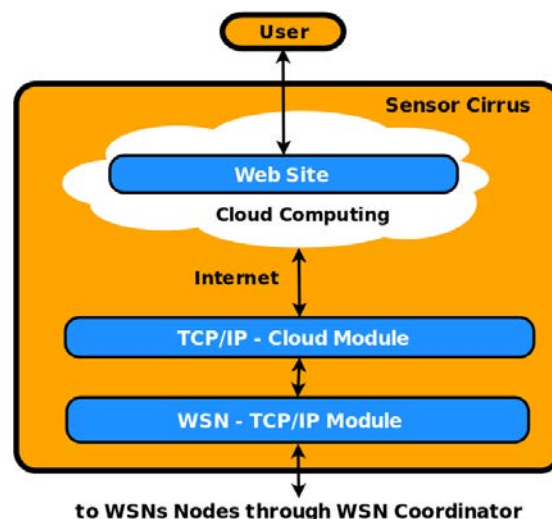

Figure 1: Sensor Cirrus Architecture Overview.

lightweight methods like TCP/IP sockets or REST services.

Even though these services have been used for WSN data share, to the best of our knowledge there are no works oriented to implement the full management of WSNs data and setup using CFSS.

3 Sensor Cirrus

In this section we present Sensor Cirrus, a platform based on CFSS for WSNs management in Sensor Clouds. This platform has been implemented and used to manage a ZigBee WSN applied to frost monitoring in crops, in Mendoza Province, Argentina. The ZigBee WSN is formed by four sensor nodes and a base station. Sensor nodes are composed by one Arduino Pro 328 [20] embedded system, an XBee - ZigBee [21] radio and a temperature and humidity Sensirion SHT15 sensor [22].

3.1 Architecture Overview

The architecture overview of Sensor Cirrus is shown in Fig. 1. It is composed by two main modules: WSN - TCP/IP and TCP/IP - Cloud, together with a website that allows the access to the platform.

The WSN - TCP/IP module of Sensor Cirrus, acts as a gateway between the WSN native protocol (in our case ZigBee) and TCP/IP. The TCP/IP - Cloud module, is composed of a CFSS service plus a so called Integration Module. The Integration Module establish the connection between the WSN - TCP/IP module and the CFSS in order to manage the WSNs data and setup. Finally Sensor Cirrus has a website deployed in a Cloud infrastructure. On the one hand, this website provides access to scripts and applications that allow the in-
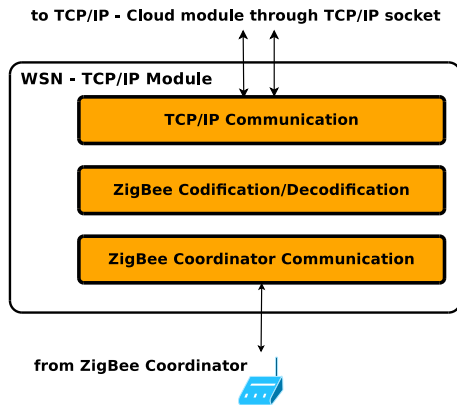
Figure 2: WSN - TCP/IP Module.

teraction with CFSS in order to display, store and process WSNs data. On the other hand, website has forms that allow the management of WSNs setup parameters like sample frequency.

3.2   WSN - TCP/IP Module

The WSN - TCP/IP module of Sensor Cirrus is illustrated in Fig. 2. This module is programmed in Python on the WSN base station, which is a PC machine running Linux Debian 7. Although WSN - TCP/IP module is programmed in python it can be migrated to c languaje and programmed on lowcost embedded systems such as Arduino platform. This module of Sensor Cirrus has been widely discussed on [19], therefore this section provides a brief description of its main components and functions.

The WSN - TCP/IP module is composed by three main processes: ZigBee Coordinator Communication, ZigBee Codification / Decodification and TCP/IP Communication. The ZigBee Coordinator Communication process connects Sensor Cirrus with the WSN Coordinator. This connection allows the reception of data from WSN and the performing of changes in sensor nodes setup. The ZigBee Codification / Decodification process extracts temperature and humidity data from the ZigBee frames and generates ZigBee frames with setup change requests.

Finally, TCP/IP Communication process establish two TCP/IP sockets with the TCP/IP Cloud module of Sensor Cirrus. One of these sockets is used to transmit the WSN data to TCP/IP - Cloud module. The other one, allows to receive setup change requests from TCP/IP - Cloud module.

3.3   TCP/IP - Cloud Module

Fig. 3 shows the TCP/IP Cloud Module of Sensor Cirrus. TCP/IP - Cloud module is programmed in a machine different from the one which host

the WSN - TCP/IP module. This is because most of the available CFSS clients (like Dropbox and Google Drive) do not provide compatibility with low-cost embedded system, which are currently used as WSN base stations. Although in this work our base station it's a PC running Linux, Sensor Cirrus must be prepared for working with base stations composed by low cost embedded systems. The TCP/IP - Cloud module has two clearly differentiated components, the Integration Module and a CFSS. In Sensor Cirrus, the CFSS used is a Google Drive Client. This CFSS is used instead Dropbox for two main reasons: the first one is that Google Drive provides more storage space than Dropbox in its no-cost version. The second one is the compatibillity of Google Drive with Google Cloud Platform [23].

It can be mentioned that the main drawback of Google Drive is the lack of portability to Linux OS. This problem can be solved using Google Drive Clients for Linux developed by other providers. However, this clients have economic costs or are still in Beta version.

The Integration Module is composed by two sub-modules (configuration and data access) and a process so called TCP/IP Communication. The Integration Module components were implemented in Python and deployed on a remote Windows PC, outside WSN. This PC contains the Google Drive Client responsible for synchronizing data and setup change request with the Cloud.
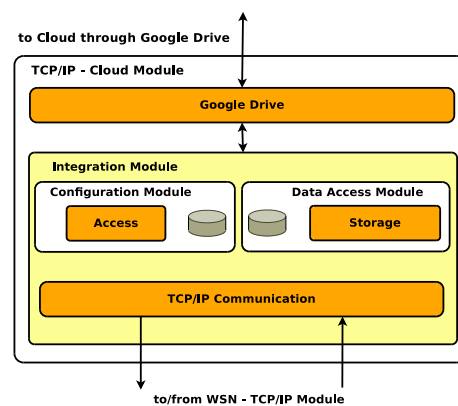

Figure 3: TCP/IP - Cloud Module.

Fig. 4 to 7 details how works the TCP/IP - Cloud module of Sensor Cirrus. First, in Fig. 4, TCP/IP Communication Process opens two TCP/IP sockets with the WSN base station. These sockets allow to receive data collected by sensor nodes and carry out changes in sensor nodes setup.
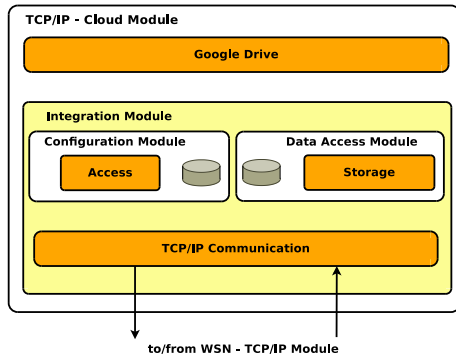
Figure 4: TCP/IP Sockets

Figure 6: Setup Change Request

After these two sockets have established the communication with WSN base station, the data access sub-module receive WSN data through Base Station and store them in files, as shown in Fig. 5. When new data come from WSN, the data files change and Google Drive synchronizes these ones with the Cloud.
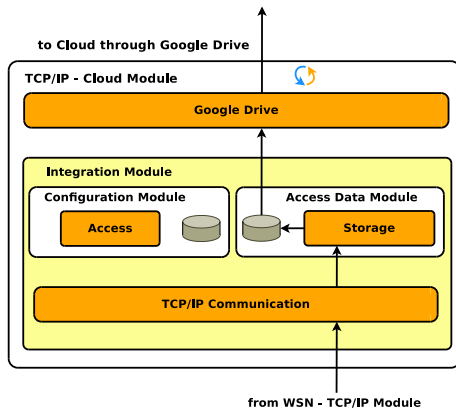
Finally, the Configuration module (see Fig. 7) through the Access process checks the last moment in which has changed the configuration file of each sensor node. If a change is registered, then a setup change must be carried out. This is performed sending the new setup value (entered by user in the website) to the Base Station through the TCP/IP Communication process.

Figure 7: Communication with TCP/IP - Cloud Module

Figure 5: Sensor Data Acquisition.

3.4   Sensor Cirrus Website

Sensor Cirrus website is the user interface of the entire platform. This website allows to visualize WSN data and perform changes in sensor nodes setup. It has been developed using the Google Sites and Google Drive toolkits. In addition, Sensor Cirrus website is hosted on Google Public Cloud. This Cloud brings hosting and access to high performance computing resources at low prices, 24 hours a day, 365 days a year.

Fig. 6 shows the management process of setup changes requests. This process begins when changes requests are performed by users through Sensor Cirrus website. Requests are stored in a WSN setup file located in a Public Cloud. This setup file is synchronized via Google Drive with the one hosted in the machine in which TCP/IP - Cloud module is deployed.
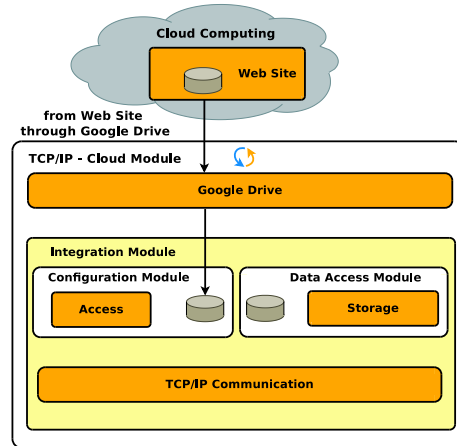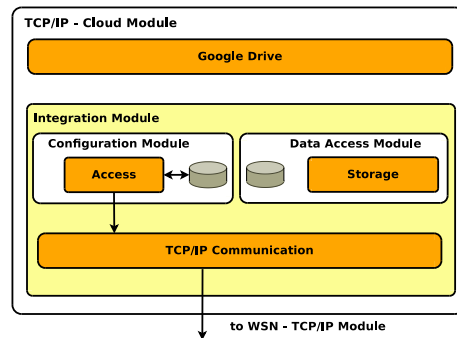
Fig. 8 details the three main sections of Sensor Cirrus website: WSN Real Time Access, Frost Alerts and WSN Setup.
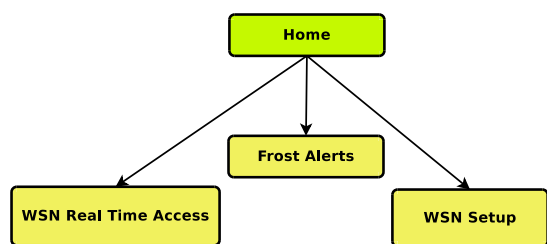
Figure 8: Sensor Cirrus Website.

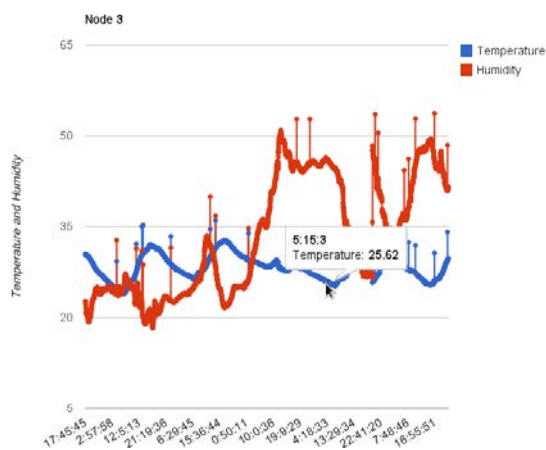WSN Real Time Access section allows the visualization of WSN data through graphics, as shown on Fig. 9.

Figure 9: Temperature and Humidity Graphics of WSN Data.

Frost Alerts section provides frost alarms, wich are emitted by a frost prediction service developed specifically for Sensor Cirrus. This service uses WSN data and Amazon EC2 instances for predict the occurrence of frosts. The frost prediction service has been widely discussed in [24].

Finally, WSN Setup section allows to perform changes in sensor nodes setup. Fig. 10 illustrates the setup change process.

First, the user accesses to Setup section in Sensor Cirrus website. Next, the user fills the following fields in a Google Drive form embedded in the website: Node in which will be performed the setup change and the new value of setup parameter. In Fig. 10 the user changes the sampling period of sensor node (this can be 1, 5, 10 or 15 minutes).

Once the form has been filled, it is sent and the information is stored in a spreadsheet on Google Public Cloud. The spreadsheet contains a script which detect the new configuration value and generates a CSV file in Google Drive. This CSV file is synchronized on the TCP/IP - Cloud module of Sensor Cirrus via Google Drive. When a change is detected in the WSN setup file, the new setup value is sent (through TCP/IP socket) to the WSN
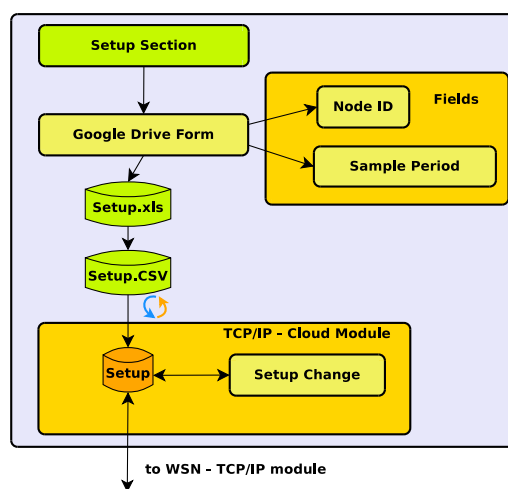

Figure 10: Setup Change through Website.

- TCP/IP module in the WSN Base Station.

Finally the WSN - TCP/IP module performs the change in sensor node through ZigBee Coordinator.

4   Experiments

This section details the experiment conducted in order to both, evaluate the performance of CFSS and determine it viability for WSN remote management. This experiment was performed using Sensor Cirrus and a WSN for frost monitoring.

4.1   Experimental Methodology

The experiment performed consists in the measurement of how many data collected by the WSN are lost in the Sensor Cirrus communication process. In order to perform the experiment, we use a ZigBee experimental WSN, which is currently applied to frost monitoring. This WSN is composed by four sensor nodes and a Base Station. Fig. 11 illustrates one of the sensor nodes of the WSN.

Figure 11: Sensor Node used in the Experiment.

The experiment is performed in two different cases. The first one is a middle-size WSN composed by 42 sensor nodes. The second one is a large-size WSN composed by 300 sensor nodes.

Currently, our experimental WSN is composed by four sensor nodes. These sensor nodes perform one data acquisition every 10 minutes. Hence, we need to use time multiplexing in data acquisition, in order to generate the same number of data frames than large-size and middle-size WSN. The time multiplexing consists in the increase of sampling rate in sensor nodes.

When data acquisition is performed in our WSN every 10 minutes, each sensor node generates 6 data packets per hour. Then, the overall WSN (composed by 4 sensor nodes) generates 24 packets of real temperature and humidity data in one hour.

In the middle-size WSN case, time multiplexing was implemented programming a sample rate of one data acquisition each 57 seconds in each sensor node. This sample rate allows the generation of 252 data frames in the overall WSN, in one hour. This quantity of frames generated in one hour in our experimental WSN is equal than the generated in a WSN composed by 42 sensor nodes performing one acquisition each 10 minutes.

The experiment in the middle size WSN is conducted during 9 days of WSN continuous functioning in laboratory conditions. Sensor nodes were deployed on the same workbench and the base station is located 1 meter from sensor nodes.

Regarding time multiplexing in large-size WSN test case, sensor nodes were programed with a sample rate of one data acquisition each 8 seconds. This acquisition rate generates 1.800 data frames in all the experimental WSN in one hour. Hence, this frame quantity is equal than the one generated by a large-size WSN, composed by 300 sensor nodes, performing one acquisition every 10 minutes.

In large-size WSN test case, the energy consumption in sensor nodes is higher than the one of the middle-size WSN. Then, the experiment only can be performed during 3 days of WSN continuous functioning because batteries were exhausted after this time period.

Table 1 summarizes each case. The second column indicates the number of sensor nodes obtained by time multiplexing and the third column the number of packets generated in one hour by the experimental WSN in each case.

Test Case	Sensor Nodes	Frames generated per hour
Middle-size WSN	40	252
Large-size WSN	300	1.800

Table 1: Test Cases.

The metric used for determining how many data collected by the WSN are lost in each link of the Sensor Cirrus communication process is the
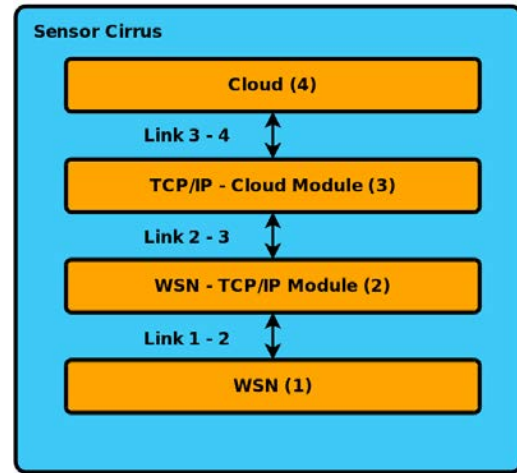

Figure 12: Sensor Cirrus Communication Process.

packet loss rate (or simply PLR). This metric is complementary to the correct packet delivery rate which was used in previous work [25] for studying ZigBee communication performance in different applications. We are going to target our studies in WSNs applied to precision agriculture, because this will be the main application of Sensor Cirrus.

Currently there is not an unified criteria for determining the correct value of PLR in WSNs applied to precision agriculture. In [26], Humber et al. develop a reliable delivery data service for WSNs. They perform studies of PLR using two environmental WSNs. The first one, its a laboratory WSN composed by 9 sensor nodes and the second one its a simulated WSN composed by 18 sensor nodes. Results in [26], shown a PLR of 7,21% in the laboratory WSN and a PLR of 11,19% in the simulated one.

In other work [27], Pierce et al. deploy in-field a WSN composed by 21 sensor nodes and obtain a PLR value between 3 and 7 %. Finally in [28] Nadimi et al. get a PLR of 8% in a WSN composed by 7 sensor nodes applied to cattle monitoring.

Based on previous works, in this paper we will consider a PLR value up to 10% for determining if the Sensor Cirrus communication process is reliable.

Sensor Cirrus communication process starts when sensor node acquires data and ends when data arrives to Cloud. Fig. 12 details the Sensor Cirrus communication process and its communication links. In Fig. 12, each Sensor Cirrus module is identified with a number and the communication links with the numbers of the modules that it connects.

Table 2 shows the extremes of each link of Sensor Cirrus communication process, the communication links and they corresponding nomenclature.

First, PLR is calculated in each link. Next, it is computed in the entire Sensor Cirrus communi-

Link	Link
Extremes	**Nomenclature**
TCP/IP - Cloud Module and Cloud	E_{34}
WSN - TCP/IP and TCP/IP - Cloud Module	E_{23}
WSN and WSN - TCP/IP Module	E_{12}

Table 2: Extremes and Nomenclature of Communication Links.

cation process. This is for determining how many packets are lost in the communication process.

The PLR value is calculated using the Eq (1):

$$PLR_{tr} = 100 - PDR_{tr} \,, \qquad (1)$$

Where PDR is correct packet delivery rate and is calculated with the Eq (2):

$$PDR_{tr} = \left(\frac{N_r}{N_t}\right) \cdot 100 \,, \qquad (2)$$

In Eq (2), t is the extreme of the link sending packets, r is the link extreme receiving packets, N_t the quantity of packets generated in t link extreme and N_r the number of received packets in link extreme r.

5   Results

This section discusses the results obtained in experiments conducted in middle and large-size WSN test cases.

5.1   Middle-size WSN

Ideally middle-size WSN generates 252 packets per hour, which represents 54.432 data packets throughout the 9 days of the experiment. This value corresponds to the 100% of packets transmitted by the WSN, then the PLR will be zero if the 54.432 data packets are received in the Cloud. However, results show that data frame are lost in some links of Sensor Cirrus communication process. Table 3 details the quantity of emitted and received packets by each component involved in the communication process.

Component	Emitted	Received
	Packets	**Packets**
Cloud	Only receives.	52.702
TCP/IP - Cloud Module	52.702	52.702
WSN - TCP/IP Module	52.702	52.702
WSN	54.432	Only emits.

Table 3: Emitted and Received Packets in middle-size WSN case.

Link	Components	PLR
E_{34}	Cloud and TCP/IP - Cloud Module	0,000
E_{23}	TCP/IP - Cloud and WSN - TCP/IP Modules	0,000
E_{12}	WSN and WSN - TCP/IP Module	3,178

Table 4: Middle-size WSN Results.

Results indicate that data lost occurs in the link between the WSN and the WSN - TCP/IP module, in this link 1.730 packets were lost. This problem is because ZigBee coordinator is busy, attending to much sensors requests. Hence, it can not send the ZigBee acknowledgment frame to all the sensor nodes. ZigBee acknowledgement frames confirm the correct reception of data and setup packets in the communication between sensor nodes and WSN Coordinator. When a ZigBee sensor node can not receive the coordinator acknowledgement frame, it retries the data transmission three times. If the acknowledgement frame from coordinator is not received by the node, then sensor node discards the data packet and enters in sleep-mode for reducing energy consumption.

In the link between WSN - TCP/IP and TCP/IP - Cloud modules all the received data were correctly delivered. This is because communication is performed over a TCP socket. Likewise, data losses were not observed in the link between the TCP/IP - Cloud module and the Cloud. This is due to CFSS (Google Drive) synchronizes all data packet with the Cloud.

The PLR values in each communication links are shown in Table 4.

It can be seen that PLR is 3,178 % in the link E_{12} , the only one that register data lost. In links E_{23} and E_{34} PLR is zero. Finally, using the packets emitted by the WSN and the ones received by the Cloud the PLR in the entire communication process is calculated. The PLR is 3,178% in the overall communication process. This PLR value shows that packet loss rate is widely less than 10% (3.178%). Therefore, we can conclude that Sensor Cirrus and it CFSS manages efficiently middle-size WSNs. It is noteworthy that our experiment was conducted in laboratory conditions, then it is expected that PLR value would be greater in outdoors working conditions.

5.2   Large-size WSN

In this section we present the results obtained in the experiments conducted in a large-size WSN. Ideally, WSN generates 1.800 data packets in one hour, which implies 129.600 data packets in the

Component	Emitted Packets	Received Packets
Cloud	Only receives.	114.748
TCP/IP - Cloud Module	114.748	114.748
WSN - TCP/IP Module	114.748	114.748
WSN	129.600	Only emits.

Table 5: Emitted and Received Packets in large-size WSN case.

Link	Component	PLR
E_{34}	Cloud and TCP/IP - Cloud Module	0,000
E_{23}	TCP/IP - Cloud and WSN - TCP/IP Modules	0,000
E_{12}	WSN and WSN - TCP/IP Modules	11,46

Table 6: Large-size WSN Results.

three days of all the experiment. Results show packet losses only in the link E_{12} of communication process. Such as middle-size WSN case, losses are due to packet traffic density on ZigBee coordinator. Table 5 details the quantity of emitted and received packets in the large-size WSN case.

In Table 5, it can be seen that packet are lost only in the link between the WSN and WSN - TCP/IP module. In this link the WSN generates 129.600 frames and the WSN - TCP/IP module receives 114.748 frames. Hence, 14.852 data frames were lost.

In the other links of communication process (E_{23} and E_{34}) all the emitted data are delivered. Like the middle-size WSN case, CFSS delivers correctly all the received data from WSN - TCP/IP module to the Cloud.

Table 6 shows the PLR values obtained on each link of Sensor Cirrus communication process in the large-size WSN case.

The PLR value for the entire communication process can be computed considering emitted packets by WSN and the ones received by the Cloud. This PLR value is 11,46% and is greater than PLR target value (10%). Obtained PLR indicates that WSN - TCP/IP Module presents problems to manage WSN composed by more than 300 sensor nodes. However, this problems could be solved decreasing the quantity of sensor nodes managed by each base station. Hence, for managing 300 sensor nodes it is recommended the use of two base stations (with 150 sensor nodes connected to each one), instead one base station per 300 sensor nodes. Another possible solution is to add more ZigBee coordinators per base station. Nevertheless, this solution should be validated experimen-

tally, studying the behavior of the USB bus or UART interfaces for the case of embedded systems used as base stations and other issues.

6 Conclusions

Currently, CFSS have opened new possibilities for the remote management of WSNs in Sensor Clouds. These services provides an easy and reliable way for WSN resources management. Among other benefits, CFSS allows reliable file synchronization, easy deployments and no specifics WSN and web services programming competences for Sensor Clouds users.

In this work we validate the use of CFSS using a platform for WSN management in Sensor Clouds called Sensor Cirrus. This platform uses a Google Drive client for file synchronization. Compared with other CFSS like Dropbox, the main advantages of Google Drive are the storage space in its no-cost version and the interoperability with other Google Cloud Services.

In order to validate the performance of CFSS for WSN management, an experiment for studying the Packet Loss Rate (PLR) in Sensor Cirrus communication process has been conducted. We studied different related works for determinate which is the PLR value of an efficient communication process in management platforms of environmental WSN. According related works, the targeted PLR value in environmental WSN communication process must be less or equal to 10% for considering the process efficient and reliable.

In the middle-size WSN case the PLR value was of 3,178%. This one is less than the targeted value. In the large-size WSN case PLR was of 11,46%. In this case, PLR value is greater than the target one. Then the platform presents drawbacks for the management of WSNs composed by 300 sensor nodes or more.

To solve the drawbacks encountered in the large-size case, there are two possible solutions. First, the quantity of sensor nodes managed by one base station must be reduced. Second, more ZigBee coordinators must be added in each base station. Hence, almost two base stations or ZigBee coordinators are necessary in a WSN composed by 300 sensor nodes or more.

Regarding frame losses in each Sensor Cirrus communication links, these ones were registered only in the link between the WSN and the WSN-TCP/IP module. This losses are due to ZigBee coordinator is busy, attending to much sensors requests. Hence, it can not send the ZigBee acknowledgment frame to all the sensor nodes. If sensor nodes did not receive this frames, after three retries discard data packets. This effect is notorious on the large-size WSNs.

The link between WSN - TCP/IP module and TCP/IP - Cloud module of Sensor Cirrus did not shows data lost. Likewise, in the link between TCP/IP - Cloud module and the Cloud all the emitted data were delivered. Hence, results show that CFSS synchronize all the data frames received in the TCP/IP - Cloud module with the Cloud.

In conclusion, from the experiments carried out and the obtained results, can be expressed that cloud file synchronization services are a reliable and efficient technology for the remote management of WSNs in Sensor Clouds.

Finally, in future works we will (i) test others CFSS like ownCloud [31], (ii) compare Sensor Cirrus with other IoT platforms and (iii) conduct experiments with the experimental WSN and Sensor Cirrus in real outdoors conditions.

Acknowledgments

References

[1] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy, "Picoradio supports ad hoc ultra-low power wireless networking," *Computer*, vol. 33, pp. 42–48, Jul 2000.

[2] L. Oliveira and J. Rodrigues, "Wireless Sensor Networks: a Survey on Environmental Monitoring," *Journal of Communications*, vol. 6, no. 2, pp. 143–151, 2011.

[3] A. Abbasi, N. Islam, Z. A. Shaikh, *et al.*, "A Review of Wireless Sensors and Networks' Applications in Agriculture," *Computer Standards & Interfaces*, vol. 36, no. 2, pp. 263–270, 2014.

[4] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2688–2710, 2010.

[5] D. Hughes, P. Greenwood, G. Blair, G. Coulson, P. Grace, F. Pappenberger, P. Smith, and K. Beven, "An Experiment with Reflective Middleware to Support Grid-based Flood Monitoring," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 11, pp. 1303–1316, 2008.

[6] M. Ghanem, Y. Guo, J. Hassard, M. Osmond, and M. Richards, "Sensor Grids for Air Pollution Monitoring," in *Proceedings of the 3rd UK e-Science All Hands Meeting*, (Nottingham, UK), 2004.

[7] S. K. Dash, J. P. Sahoo, S. Mohapatra, and S. P. Pati, "Sensor-cloud: Assimilation of wireless sensor network and the cloud," *Advances in Computer Science and Information Technology. Networks and Communications*, pp. 455–464, 2012.

[8] K. Ahmed and M. Gregory, "Integrating Wireless Sensor Networks with Cloud Computing," in *Seventh International Conference on Mobile Ad-hoc and Sensor Networks (2011 MSN)*, pp. 364–366, IEEE, 2011.

[9] M. Hirafuji, H. Yoichi, T. Kiura, K. Matsumoto, T. Fukatsu, Tanaka, and Others, "Creating High-performance/Low-cost Ambient Sensor Cloud System using OpenFS (Open Field Server) for High-throughput Phenotyping," in *Proceedings of 2011 SICE Annual Conference (2011 SICE)*, pp. 2090–2092, IEEE, 2011.

[10] M. Hori and T. Kawashima, E.and Yamazaki, "Application of cloud computing to agriculture and prospects in other fields," *Fujitsu Scientific & Technical Journal*, vol. 46, no. 4, pp. 446–454, 2010.

[11] Google Drive: Available on: https://www.google.com/drive/.

[12] Dropbox: Available on: https://www.dropbox.com/.

[13] K. Lee and D. Hughes, "System Architecture Directions for Tangible Cloud Computing," in *International Workshop on Information Security and Applications (IWISA 2010)*, vol. 25, (Qinhuangdao, China), Octubre 2010.

[14] K. Lee, D. Murray, D. Hughes, and W. Joosen, "Extending Sensor Networks Into the Cloud Using Amazon Web Services," in *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA)*, pp. 1–7, IEEE, 2010.

[15] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[16] Xively a Public Cloud for the Internet of Things: Available on: https://www.xively.com/.

[17] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V. Leung, "Lightweight management of resource-constrained sensor devices in internet of things," *Internet of Things Journal, IEEE*, vol. 2, no. 5, pp. 402–411, 2015.

[18] M. Navarro, T. W. Davis, Y. Liang, and X. Liang, "A study of long-term wsn deployment for environmental monitoring.," in *PIMRC*, pp. 2093–2097, 2013.

[19] L. Iacono, *Gestión Remota de Redes de Sensores Inalámbricas Mediante Tecnologías de Cloud Computing*. Phd thesis, Universidad de Mendoza, October 2015.

[20] Arduino Open Source Hardware: Available on: `https://www.arduino.cc/`.

[21] XBee ZB Modules Data Sheet: Available on: `http://www.digi.com/`.

[22] Sensirion SHT15 Data Sheet: Available on: `http://www.sensirion.com`.

[23] Google Cloud Platform: Available on: `https://cloud.google.com/`.

[24] L. Iacono, J. L. Vázquez-Poletti, C. García Garino, and I. M. Llorente, "A model to calculate amazon ec2 instance performance in frost prediction applications," in *High Performance Computing*, pp. 68–82, Springer, 2014.

[25] P. Godoy, L. Iacono, R. Cayssials, C. Párraga, and C. García Garino, "Effect of working conditions over the performance in ZigBee WSN," in *Capítulo Sistemas de Control, Anales de la primera reedición del Congreso de la Sección Argentina del IEEE (Argencon 2012).*,
(Córdoba, Argentina), 2012. ISBN: 987-572-076-3.

[26] G. Humber and E. Ngai, "Quality-of-Information Aware Data Delivery for Wireless Sensor Networks: Description and Experiments," in *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pp. 1–6, IEEE,2010.

[27] F. Pierce and T. Elliott, "Regional and on-farm wireless sensor networks for agricultural systems in Eastern Washington," *Computers and electronics in agriculture*, vol. 61, no. 1, pp. 32–43, 2008.

[28] E. Nadimi, H. Søgaard, T. Bak, and F. W. Oudshoorn, "Zigbee-based wireless sensor networks for monitoring animal presence and pasture time in a strip of new grass," *Computers and electronics in agriculture*, vol. 61, no. 2, pp. 79–87, 2008.

[29] R. Beckwith, D. Teibel, and P. Bowen, "Report from the field: results from an agricultural wireless sensor network," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pp. 471–478, IEEE, 2004.

[30] S. E. Díaz, J. C. Pérez, A. C. Mateos, M.-C. Marinescu, and B. B. Guerra, "A novel methodology for the monitoring of the agricultural production process based on wireless sensor networks," *Computers and Electronics in Agriculture*, vol. 76, no. 2, pp. 252–265, 2011.

[31] ownCloud: Available on: `https://owncloud.org/`.