StoNes – Multiprotocol Gateway for Wireless Sensor Networks

Rui M. L. Santos, Instituto Superior Técnico, Universidade Técnica de Lisboa

Abstract — The StoNes system fills the gap left by the current WSN, with respect to the interaction between them, as independent WSNs. The central piece of this system is the StoNes Gateway, which is implemented using a modular programming architecture, with independent processes that manage each WSN, creating a robust system and an easy method to add new WSNs. The common concerns associated with these networks are also addressed, such as mobility of nodes, communication failures, heterogeneity of nodes, unattended operation and ease of use. One of the examples concerning ease of use is the hostname based addressing for the sensors within StoNes, allowing a simple way to interact with the sensors using a web interface.

In contrast with common solutions, where each WSN works independently, this architecture allows the creation of heterogeneous WSN environments, not only regarding the sensors diversity but also the diversity of WSNs working together, aggregating different protocols into an organized scheme, where the StoNes Gateway is responsible for managing those networks. Each process, responsible for each WSN, only translates the WSN message protocols into the one used internally by StoNes.

The solution was implemented using standardized options so that the software could be executed in several different systems. It has successfully been tested on multiple hardware solutions, from an High-End Desktop running Ubuntu or a Macbook Pro with OSX Lion to a Bifferboard running Debian. The system was tested using the Bifferboard and the doBros WSN with positive results, capable of coping with all situations covered by that WSN.

Index Terms — WSN, Integration, Heterogeneity, Remote Access, Secure Access.

I. INTRODUCTION

With the constant evolution of microcontroller technology, whether in cost, size and functionality, WSN are becoming more popular as a mean to acquire data and an easy method to place sensors in places with limited access. With increase demand of Home Automation solutions, wireless sensors and actuators are under a constant evolution. These solutions may differ in the adopted technologies and architectures or the type of sensors and elements they act upon, but one thing they have in common, a system from one developer usually cannot be integrated with the system from other.

The main objective of this work is to create a system, StoNes, that could integrate independent WSNs (in a true heterogeneous environment) into one centralized gateway, an abstract layer when dealing with data transmission, but providing a simple and efficient method to interact with sensors to configure or even control them.

Considering the referred architecture some requirements were considered important, for the envisioned system.

- Overall system robustness/reliability;
- Modularity/easy WSN integration;
- Multi-Platform;
- Good performance;
- Security;
- Scalability.

II. STATE OF THE ART

This section will compare existent WSN solutions targeting environments in the StoNes scope, in architecture/model and analyze how they compare regarding technology, architecture and security.

A. Protocols

The most common wireless technology is IEEE 802.15.4 [1], which is known as the standard protocol for WSN, it specifies the physical layer (PHY) and the media access control (MAC) layer for low-rate wireless personal area networks (LR-WPANs) and being the support to higher level communication protocols, such as ZigBee [2], 6LoWPAN [3] or the NXP JenNet [4]. It allows data rates up to 250 kbps, two addressing modes (16-bit short and 64-bit IEEE addressing) and automatic network establishment by the coordinator, power management to ensure low power consumption and transmission on the 2.4 GHz ISM, 915 MHz and 868 MHz band.

Other popular wireless standard is the IEEE 802.11 [5], although not originally designed for WSN, higher bandwidth requirements than the ones provided by traditional protocols, may be found in these networks (e.g.: video security feeds). It operates in the 2.4 GHz, 3.6 GHz and 5 GHz frequency bands, with a range up to 250 m (802.11n), with bandwidth up to 150 Mbit/s (802.11n).

Bluetooth [6] is the most common used protocol for PAN, for communication between mobile phones, PDA and computers. The latest version, Bluetooth Low Power (BLE) (2010), when compared with Bluetooth v2.1 (2007), comes
with an increase in range (from 30 m to 50 m), a decrease in bit rate (1-3 Mbit/s to 200 Kbit/s) and a decrease in power from 50% to 90% by optimizing three basic areas of functionality: connectable and discoverable modes, the number and size of the packets transmitted during connections.

Z-Wave [7] is a proprietary protocol developed by Zensys, providing mesh networking and designed for home automation. Its operation band is around 900 MHz, which guarantees less interference than, for example, Wi-Fi (2.4 GHz). It is yet another communication protocol for WSNs.

DASH7 [8] is a WSN protocol, following the ISO/IEC 18000-7 standard [29], operating in the 433.92 MHz band, providing data rates up to 200 Kbit/s, and advertising ranges up to 10 Km and a propagation signal capable of penetrating walls, concrete and water, providing a more versatile solution than the ones aforementioned, concerning working environments.

ZigBee [2] is a network standard protocol built over the IEEE802.15.4 working as a complement, adding routing and networking functionalities such as mesh capability, and being intended for embedded systems, providing at the time of creation, an alternative to unfitted (at the time) solutions such as Wi-Fi and Bluetooth.

The last one to be mentioned is 6LoWPAN [3], the application of the Internet of Things concept to WSNs, by the application of IPv6 to sensors. It defines header compression schemes and data fragmentation methods for the transfer of IP frames over the limited frame of IEEE802.15.4 that is the support protocol for 6LoWPAN. IP protocol integration in WSNs has the advantage of using a well-established protocol with a vast addressing space (IPv6 vs IPv4) and solid operating procedures.

B. Hardware Platforms

The first one to be mentioned is the NXP Jennic JN511XX [4] platform, used in the doBroS WSN, used for testing StoNes. These modules, developed by NXP Jennic, run a proprietary stack over IEEE802.15.4. Among the supported protocols there are 6LoWPAN and ZigBee stacks and a proprietary Jennic communication protocol, the JenNet. It should be referred however that the 6LoWPAN and ZigBee are not compliant with the current standard versions.

The second platform is the MIT Environmental Sensors or MITes [13], with wireless sensing devices, optimized for easy use, installation, affordability and robustness. Its sensors include movement, light, temperature, proximity, and current sensing in electric appliances in a scheme of a network with a central gathering information node (coordinator). This is the same type of network used by Jennics JenNet.

The Mica Mote [14][15] is probably one of the most famous modules used for WSNs. Built from a partnership between Berkley University of California and Crossbow, Inc. was one of the first platforms built for WSNs. Is composed of a microcontroller and a radio, and supports TinyOS [16], an OS designed specifically for embedded systems, by that same University. Many versions of the Mote exist, among them Mica, Mica2, Mica2Dot, MicaZ, Telos and IRIS. Du to its popularity, other motes followed the same platform, like the BTnode [17][18]. But this device has Bluetooth radio and a microcontroller, and is developed at ETH Zurich. Since it shares Mica Mote platform it has similar functionalities and is TinyOS programming compatible, making it an alternative to Mica.

The Intel® Mote 2 or iMote² [19] is the Intel approach to WSN. Integrating a IEEE802.15.4 radio. Supporting a variety of OS, like TinyOS, increases the versatility of the mote. The work using a proprietary protocol, with the nodes organized in a tree like architecture with a central node for central data processing.

C. WSN Existent Implementations

WSNs have innumerable application scenarios, from military applications [9], to monitor the environment [10], health care [11], security, residential control, etc., acquiring data from several types of sensors and interacting with the devices, such as the example of the Home Automation solutions, with the sensors and actuators providing the users with a better “in home” experience.

This is the case of examples like Control4 [21] with its home control and automation solutions, in the same line as Home Automation, Inc. [22] or the Portuguese Life Emotions [23] with “Kasa do Futuro” concept, or projects like MavHome [24], Aware Home [25], MIT House_n [26] (with the MITes Motes, mentioned before), Gator Tech Smart House [27], UbiHome [28], the Domo Zonse [12] project or the doBros [11] project.

D. Beyond State of the Art

The aforementioned examples are valid for residential scenarios, but what if the scenario is not a residence, but an aggregation of houses, a building or even multiple buildings?

For instance, a university campus, like Instituto Superior Técnico or a research institution like INESC-INOV.

The objective behind StoNes is not only to idealize, but also to be able to go from concept to a prototype, applying those same concepts.

Leaving behind the limitation of a residential home, facing larger environments.

Figure 1 - From the residential to the large environment.

StoNes aims to be an aggregator gateway for heterogeneous WSNs, but the heterogeneity of the sensors isn’t the only concern, as referred in MARWIS [31]. There are other concerns regarding the network, allowing the coexistence of
different communication protocols, and promoting the creation of hybrid solutions, to better cope with the necessities of the installation setting. For example, a Bluetooth Network that acquire temperature information, and a different network with ZigBee responsible for the irrigation systems, but with a single shared interface, with easy access to the sensors, but guaranteeing secure data transmission and an easy sensor/actuator addressing method.

E. \(\mu\)PNP(v2) and doBros WSN upgrade

The StoNes architecture implements a newer version of the original \(\mu\)PNP message protocol, called \(\mu\)PNP(v2). These modifications expand the protocol potential, by improving the \(\mu\)PNP header format, more specifically on the message types and their division. This will be explained in more detailed in the next chapter.

The first modification to the doBros WSN was to update the \(\mu\)PNP format to the new \(\mu\)PNP(v2). Taking advantage of the modularity of doBros, only the update of the \(\mu\)PNP library was necessary.

The other important change is the suppression of the Sensor Proxy, since its purpose, to connect doBros to the Internet, is now carried out by the StoNes Gateway, that works as an aggregator of all elements (Internet, Users and WSNs). The doBros WSN coordinator in the StoNes system is simple JN5139 module connected to the StoNes Gateway. This corresponded to a change in the coordinator source code, removing the NanoSocket [37] [38] library and adding the functions necessary to communicate using the serial port via the USB connection.

III. StoNes

Designed and developed as a modular system, and using a multi-process architecture for flexibility and robustness, takes advantage of Message Queues [32] for inter-processes/threads communication. The use of \(\mu\)PNP [11], DTN, sensor identification and addressing and processes division, network access were all things taken into consideration on building the system, represented on Figure 2.

A. StoNes Architecture

1) Overview

The most important component of StoNes is the StoNes Gateway. It is the central piece of the system, where the WSNs coordinator/sink node is connected, it is the element that connects to the exterior, and it is also responsible for the User interfaces.

The StoNes Gateway is the “middleman” between User/Server and the WSNs, but its true feature is to pass undetected on User↔Sensor communication. The system is implemented in a transparent way, such that there is no real notion of a gateway and as such must be data-agnostic.

The gateway was separated in 3 main blocks, each one with a very specific purpose:

1. The WSNs Coordinator interface, responsible for managing the communication interface that is connected with the WSN coordinator.
2. The Internet interface, dealing with the connections from and to the outside of the gateway.
3. The main block, called Controller.

There is a separation from the Controller to the elements that may suspend normal program execution, like reading from serial port or from sockets. This separation if represented on Figure 3, with the Controller (process), managing the interfaces.

2) Sensors Identity

Sensors are perceived as equal members of a unique WSN, equal among them and directly addressed via its own hostname, even thought they can belong to different and independent WSN, use different communication protocols.

Comparing IP vs Hostname, the IP one showed to be more difficult. Using a virtual hostname is more interesting and adequate in this type of architecture, when the gateway is the evident manager of the communications. They are virtual hostname because they all refer to a single IP address, the Gateway where the WSN is connected.
And it is easier to memorize a name that a number (in IPv4 they are four numbers, in IPv6 there are 8) and if this name can be configured by the user, which is the case in StoNes, and from that point of view, it made more sense to adopt a hostname naming system.

The initial hostname addressed to the Sensor by the Server is based on the Sensor UID.

3) Delay-Tolerant Network (DTN)

In a WSN one of the main characteristics of the sensors is the need for low maintenance. So, robust hardware and high battery life are among the main concerns. The best way to conserve battery is to put the sensors in a low power (sleep) mode whenever they are inactive. When sleeping, sensors usually turn off the radio (the most battery draining element) making them cut off from the network during those periods.

The problem of contacting an element of the network, which is momentarily in sleep mode is being researched in the DTN Research Group [33] as part of the Internet Research Task Force (IRTF) [39].

StoNes takes this into account by passing all communication thought the Controller and not sending it directly to the sensors. In the event that a sensor is unavailable, the “communication manager” delays or resends those messages to the sensor and gives feedback to the element that made the request (e.g.: user, server) about the current state of the sensor, current problems, expected time to the next window of availability and so on.

B. μPNP (v2)

To cope with the designed architecture, and considering the inherent limitations of WSNs and its common hardware, a single message format is used in StoNes Gateway.

From the available formats, the choice went to a variation of the UPnP [20] developed for the doBros [11] project and named μPNP [11], which is a WSN oriented format, simple in message format and has small overhead was considered the optimal solution, using the KLV encoding standard for data and having the message integrity verified with the use of a checksum.

The format is very straightforward and its implementation easily achieved. It uses a scheme of Dictionaries with the information regarding each sensor, with a scheme of keys for each parameter, ending with a communication with a low message overhead and direct translation method. Adding new sensors is also simplified by the use of Dictionaries, in the XML format, that can make this architecture and system even more versatile. The variation to the original μPNP is related with the Header format, modified to better suit WSN needs, by updating the message types and data classifications.

The μPNP Dictionaries can be separated in two types, the Common Dictionary, with the common parameters to all WSN and the Specific Dictionary, with keys specific to each sensor. Each element (each existent key) of these Dictionaries is composed of 5 parameters:

- Key: e.g. DICT_COMMON_KEY_UID;
- Type: e.g. KLV_TYPE_CONFIG_RO;
- Format: e.g. KLV_FORMAT_BINARY;
- Description: e.g. “uid”;
- Options: \{[-1, ""]\};

The Key is the element identifier, that also distinguishes between common or specific, the Type corresponds to the type of parameter, if it is a configuration, data or request, the Format of the data, the Description that will be presented to the User and finally the values supported by that parameter.

<table>
<thead>
<tr>
<th>Bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Request</td>
<td></td>
<td>Data</td>
<td>Type</td>
<td>Cache</td>
<td>Long</td>
<td>Encrypt</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Payload Length</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Offset (Optional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>up to 128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>Payload</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chaesum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There were considered 4 focal message exchanges that brought forward the design of the protocol as it is. Beginning with the Sensor Advertise, exemplified in Figure 4, where there are two situations, a successfully POST to the DDNS (1) and a response with a hostname, and the event that the DDNS server is unavailable and not hostname can be assigned (2).

![Figure 4 - Sensor Advertise Request and Server Response (message exchange).](image)

![Figure 5 - Sensor Advertise Request and Server Response (data exchange).](image)
Table 2 - Sensor Advertise and Server Response (data detailed).

<table>
<thead>
<tr>
<th>Sensor Advertise Request</th>
<th>DDNS Server/Advertise Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>µPnP Header</strong></td>
<td><strong>µPnP Payload</strong></td>
</tr>
<tr>
<td>Type and Flags: 00101001</td>
<td>Key: 0x03 - Sensor Type</td>
</tr>
<tr>
<td>Req. No.: 0x80</td>
<td>Length: 0x10</td>
</tr>
<tr>
<td>Payload length: 16</td>
<td>Value: 0x80 0x64 0x31</td>
</tr>
<tr>
<td></td>
<td>Key: 0x0A - Hostname</td>
</tr>
<tr>
<td></td>
<td>Length: 0x10</td>
</tr>
<tr>
<td></td>
<td>Value: 0x62 0x2D 0x31</td>
</tr>
<tr>
<td></td>
<td>Key: 0x0A - Destination</td>
</tr>
<tr>
<td></td>
<td>Length: 0x10</td>
</tr>
<tr>
<td></td>
<td>Value: 0x62 0x6D 0x20</td>
</tr>
<tr>
<td></td>
<td>Key: 0x0A - ID</td>
</tr>
<tr>
<td></td>
<td>Length: 0x10</td>
</tr>
<tr>
<td></td>
<td>Value: 0x62 0x73 0x6D 0x30</td>
</tr>
<tr>
<td></td>
<td>Key: 0x0A - Message</td>
</tr>
<tr>
<td></td>
<td>Length: 0x10</td>
</tr>
<tr>
<td></td>
<td>Value: 0x62 0x6D 0x30</td>
</tr>
<tr>
<td></td>
<td>Key: 0x0A - Access</td>
</tr>
<tr>
<td></td>
<td>Length: 0x10</td>
</tr>
<tr>
<td></td>
<td>Value: 0x62 0x6D 0x70</td>
</tr>
<tr>
<td></td>
<td>Key: 0x0A - Acknowledgment</td>
</tr>
<tr>
<td></td>
<td>Value: 0x62 0x6D 0x10</td>
</tr>
<tr>
<td></td>
<td>Key: 0x0A - Request</td>
</tr>
<tr>
<td></td>
<td>Value: 0x62 0x6D 0x30</td>
</tr>
<tr>
<td></td>
<td>Key: 0x0A - Response</td>
</tr>
<tr>
<td></td>
<td>Value: 0x62 0x6D 0x70</td>
</tr>
</tbody>
</table>

In the DDNS Server response, the assigned hostname is 00158D000013847.ohm.inesc-id.pt.

The second situation to be considered was Request Message with the sensor measurements, to be sent to a remote Database.

In the event of a Sensor configured with measure intervals largely spaced, a Keep Alive Mode is implemented in the protocol, so that from time to time, it would wake up, inform StoNes of its status and receive configuration/request if necessary.

![Sensor Keep Alive message exchange](image)

When the sensor communicates with the Gateway, it creates a window of opportunity to communicate with it and it is in this frame that configurations are update and values requested, just like it was a Keep Alive Mode.

If the sensor is not available, all requests are cached until they can be sent.

C. StonesGateway (Building Blocks)

1) Controller

As mentioned before this is the StoNes Gateway main process, manages the communication between all the processes and threads of the gateway using a system of Message Queues. They are very simple to implement and using a FIFO scheme that guarantees order on message reception and delivery being at the same time a non-blocking communication method, and allowing the use of custom designed structures with the desired size.

Controller has 2 message queues:

- RES – Response Message Queue;
- REQ – Request Message Queue.

This division into two queues allows the implementation of a priority organization of the message exchange, providing a better way to organize them, giving a higher emphasis to Sensor Requests.

This process also contains a record of the sensors that are connected to the Gateway, keeping that data on a list. This list contains all the pertinent data concerning each sensor. This is facilitated by the fact that every communication inside the StoNes Gateway passes through this process.

2) WebClient

These threads are responsible to POST sensor information on the servers (from measurement values to the remote Database to making update requests to the DDNS Server). They receive the message from the sensor in µPnP, convert it to JSON [34] and send it using the POST method. But, on the event of a problem in the connection with the Database, whether the Database is offline or the Gateway itself lost Internet connection, it create backup of not sent data in a file. If, on the other hand, that message is an Advertise message to the DDNS server that could not be delivered, it returns an ACK to the sensor and it is at this time the status field on the Sensor is set to NotRegistered until the connection is reestablished.

If the sensor data were to be sent to other services (different from the StoNes Database), and using any other data format, for example pachube [35], it was only necessary to change the translation of the data to the library designed for pachube, keeping the code of the WebClient almost unchanged.

3) WebServer

This is the interface dealing with external requests, in the format of a webpage to the User.

When a new connection is initiated, a WebServer thread launched to deal with this request, taking ownership of the connection from the main process. It responds by giving to the user a webpage with all the information available for the requested sensor. This webpage is generated on the fly, specifically for the requested sensor, making use of the Dictionaries and all the info present in Sensor List. The webpage is a very simple one, using the jQuerys to request data in JSON format, and present it creating a visually attractive and simple layout, and transferring the processing weight to the User browser. This data conversion from µPnP to JSON is achieved by the use of the Dictionaries.

4) Agents

The Agents act as interfaces from the WSN to the StoNes Gateway, responsible for managing communication (e.g.: Serial Port, SPI, etc) and data translation to and from µPnP, in a completely stateless mode, since they keep no data records. One of the Agents is the doBros-Agent.

This Agent then combines the communication with the doBros-GW, through a serial port, using RS-232 protocol and to guarantee message integrity in the serial communication, a scheme with control characters was implemented on both sides of the communication line, using SOH (start of heading), STX (start of text) and ETX (end of text).

![Example of a Serial communication](image)

In doBros-WSN, identify each sensor the sensors MAC Address (8 bytes) is used, guaranteeing a unique ID, and the initial hostname assigned to that sensor.
Other case is the RFID-Agents. The CAEN RFID reader has up to 4 antennas, and each antenna is treated as if it is a single RFID reader, so an individual sensor, and to each antenna is assigned its own hostname and can be configured independently, being the hostname a result of, for example, geographic location.

Even using the StoNes system with WSN systems based on 6LoWPAN could be useful for the same reason. It allows a single look and consistent interface to the user, hiding all the complexity in the StoNes gateway, so that the end-nodes do not need to implement their own webservers for communication, but rely on the stones gateway to do the heavy lifting.

IV. EVALUATION

This chapter evaluates StoNes, providing a performance and functional assessment of the system in various test scenarios, to infer about the suitability of the implemented architecture design and programming options and choices against the ordinary concerns regarding WSNs and their default environments, and the objective of the developed system in the proposed working scope.

An evaluation on the system functionality parameters is made, such as response to failures and overall reliability mechanisms, remembering that some of the WSNs internal malfunction and failures cannot be pinned on StoNes.

The system response time will be evaluated on a standard environment, considering one scenario with one sensor and one with multiple sensors.

A. Functional Evaluation

This section will analyze test scenarios from a functional point of view, using the doBros WSN as the testing WSN. So regarding the expected failures, the next evaluation will provide some insight to the adopted solutions concerning functional problems, namely lost messages, offline servers or the inexistence of Internet connection and also security related issues.

1.1.1 Offline Servers

The first example refers to a failed connection to the Database, which can happen in one of two cases: inexistence of Internet connection or Database is offline. Independently of the case of the messages not being delivered (interrupted POST) or undeliverable (Server if offline or no Internet connection), it is saved locally in file until the connection is established, time when StoNes provides de Database with the URL of the saved file, leaving the decision of retrieving it, to the Server.

The case of a problem with DDNS Server is more difficult to deal with, once without it, the sensor cannot be addressed.

1.1.2 Security

An important issue in this work is security related. To deal with it, it’s used a VPN that encloses the entire system StoNes, Users, Servers.

This way, only authorized users and StoNes Gateways have permission to interact with both Servers (Database and DDNS) via the POST method and access to the saved data. The same is valid for the Sensors web interface via the hostname, since the DDNS server, also inside the VPN, provides the DNS. The idea of having a system behind a VPN also expands the liberty on choosing the type of Internet connection to be used. A StoNes Gateway can connect using an open Wi-Fi connection and still guarantee a secure data transfer and limited and verified access to the Sensors. The same is valid for a 3G connection or any other type.

B. Performance Evaluation

In this section the system performance is scrutinized, analyzing the message exchange between the system elements and StoNes response to external (User or WSN) requests and...
the influence of the increase on the number of Users accessing StoNes or the number of Sensors connected to it. This evaluation will also compare the system performance, with StoNes running on an Intel Core i7 (Quad-Core Processor @ 3.0 GHz) Desktop versus the Bifferboard (S3282 BGA CPU @ 150 MHz) [36].

The normal working scenario for doBros is considered to conduct this performance evaluation. This scenario is composed of several spontaneous measurements from one or multiple sensor and the same situation but with Users accessing the website, increasing the load on the system. The message propagation diagram is represented in Figure 12.

![Message propagation diagram](image1)

Figure 12 - Message propagation diagram.

This is the progression of a message Sensor originated from a doBros Sensor, and the corresponding Server response. The time from the moment it is available on StoNes Gateway (is read in the Serial Port) until the time the Server response is delivered to the WSN is considered, taking into account the processing time and the waiting time on the queues.

![Average message propagation time](image2)

Figure 13 - Average message propagation time, split by elements for a single Sensor. (Bifferboard)

The previous figures represent the average time necessary for a message originating from a Sensor to reach the destination Server and return from the WSN Coordinator to the Sensor. These charts represent that time for a single Sensor, and using the Bifferboard as the StoNes Gateway.

The same scenario, but while Users are requesting information from the system (browsing the sensor website) is represented in Figure 15. Since the StoNes Gateway has to process more requests, an increase in time is noticed on the network element of the system (the User requests increase the load in that component) and in the Serial Port communication.

![Average message propagation time](image3)

Figure 14 - Average message propagation time, split by processing and queue time for a single Sensor. (Bifferboard)

The next test scenario uses the same WSN but now having multiple sensors connected.

![Average message propagation time](image4)

Figure 15 – Average message propagation time split by elements, with and without simultaneous User requests.

![Average message propagation time](image5)

Figure 16 - Average message propagation time split by elements, for the 3 individual sensors (Bifferboard).
In this situation a decrease in system performance is noticed. It is more noticeable for the case of Sensor 2 and 3, which were initialized at the same time and were transmitting data almost simultaneously.

The Desktop was subject to the same test scenarios and the results presented below.

For the single Sensor the performance is much better than the one from the Bifferboard, like shown in Figure 19. Curiously, the queue-time is almost the same (108 ms vs. 89 ms), whereas the processing time for the Desktop is much smaller (6 ms vs. 98 ms).

For the multiple sensors case the Desktop does not suffer visible performance loss, with times around 120ms, as shown in Figure 21 and Figure 22.

With average message times almost half of the ones experienced in the Bifferboard (Figure 23) but still in the order of the tenths of a second, the Bifferboard is a little bit slower.
With User requests, the Desktop does not suffer significant loss in performance so no charts pertaining that situation were added in this evaluation.

Regarding the system response to User request an evaluation was made from the User point of view. The time necessary to respond to a parameters request (user browser update information every 2s, requesting new data) and to update the latest sensor measurements (user browser updates latest sensor data every 30s) is analyzed.

For each request, Stones creates a new JSON file on the fly. And since the webpage uses JavaScript and jQuery to send only the updates, it lightens the load on the WebServer, transferring the layout to the user web browser.

As observed in Figure 24 the response time of the WebServer running in the Bifferboard for Sensor measurements in case of single and multiple users is very similar. The first case with a mean value of 50ms and a standard deviation of 9ms and the other with a mean value of 51 ms and a standard deviation of 14 ms. It is similar for the access to the Sensor information, with a mean value of 78 ms and a standard deviation of 4 ms and the other with a mean value of 92 ms and a standard deviation of 21 ms.

As expected, the overall performance of the Bifferboard is inferior to the one of the regular Desktop PC. But the difference was not as big as expected, considering the gap in the hardware capabilities, especially for the case of a single sensor. For multiple sensors, the Bifferboard falls behind with values almost double to the one in the Desktop PC. Even so, those are values in the tenths of a second, which is very acceptable for the User.

V. CONCLUSIONS

The StoNes approach is able to deal with the original ideas and concerns regarding ordinary WSN concerns, plus the important concepts such as DTN. A model capable of dealing with the main points of interest associated with the WSN is achieved, namely, the achievement of integrating different WSN creating true WSN heterogeneity, at the same time running a robust system with very interesting features.

The most interesting one is hostname based sensor addressing, an easy and intuitive way to access the sensor, even if in reality the sensor may not be accessed directly or in real time. But to the User that conception is smoothed by the caching ability of the system, providing the User with the latest information of each sensor, ordinarily enough, not needing to burden the network with unnecessary traffic.

To connect all the elements, Stones Gateway, Servers and Users and to guarantee secure data transfer to limit access to authorized users, a VPN is used. This allows every element inside the VPN can be accessed from anywhere in the world and geographically separated WSN share the same unified platform, like they were in rooms side by side (the example of both IST campus, Alameda and Tagus, separated by several kilometers, but inside the same network, controlled like if they were the same location).

The message protocol, with the choice of µPNP(v2), an update from the original µPNP, but more versatile and with new functionalities, fit all the system needs, easy to use if need to be implemented and if not, easy for the Agent to translate to and from.

REFERENCES
