Multi-layer architecture for location systems based on wireless sensor networks

Javier Rodas, Carlos J. Escudero
Departamento de Electronica y Sistemas. Universidade da Coruña.
A Coruña. Spain
jrodas@udc.es, escudero@udc.es

Abstract—Location systems based on WSN are built using diverse technologies. Factors such as the programming language to implement the location algorithm, the underlying operating system (available libraries, closed APIs,...) and type of hardware node interfaces, frequently drive us to a non-reusable design. Usually, the WSN technology cannot be easily replaceable by another one. We introduce a novel layered architecture that allows the abstraction of both WSN and location algorithm layers, communicating them by means of XML-RPC. As main advantages, both layers are now reusable, can be implemented with different programming languages, be distributed over different machines and it is possible a concurrent access to the WSN from different clients.

I. INTRODUCCIÓN

The design of a location system based on WSN [1] often arise doubts about the technology and the software that will be used. The technology used for the sensor network (WiFi, Bluetooth, ZigBee, etc.) is often set in the first place, which severely restricts the design, due to the particularities of each one to obtain measurements, their interfaces of communication, etc. Some constraints usually arise due to software drivers and libraries available on the selected operating system, which is running on the host that collects and processes the measurement data. Open operating systems usually allow a deeper and freer low-level access, so they are preferable when implementing these systems. Another limitation is given by the programming languages in which the libraries (or APIs) needed to access the WSN hardware are implemented. Therefore, location systems are usually implemented using a single programming language and operating system (the one which gives us maximum flexibility). Typically we are also forced to rewrite the localization algorithm (sometimes quite complex), which we already had programmed and tested using high-level simulation environments like Matlab. This task would often requires a great cost and time. Furthermore, when we need to get data from the WSN in real time, we are forced to adapt the location algorithm in order to work with stream-type data. This usually requires an enormous effort for some programmers who usually only work with off-line data (log files or simulated data).

In this paper, we introduce a new architecture that minimizes the number of restrictions and limitations we usually suffer when implementing a location system. We have divided it mainly into two major layers:

- **Physical**: It is responsible for abstracting the way we access the WSN hardware to collect the measurements.
- **Location**: It implements a location algorithm that is fed with data from the previous layer.

Both layers become independent and reusable. We can change the technology used in the sensor network without changing the *Location* layer, or change the location algorithm without altering the *Physical* layer.

To communicate both layers, we use the XML-RPC standard protocol, a remote procedure call (RPC) protocol that adds many additional advantages to the system:

- Being an open specification standard, it is implemented in all kind of programming languages and operating systems (multi-platform). It is possible to implement the *Physical* layer (which accesses the hardware) using low-level languages and libraries (typically programmed in C), while another higher-level programming language (Java, Matlab, PHP, C++, etc.) can be used to implement the *Location* layer.
- Both layers can run on a single machine or on different distributed machines, interconnected by an intermediate network or the Internet. Data is formatted in XML and transported using HTTP and, therefore, they pass through any firewall or subnet without opening ports.
- It is a client-server architecture, where concurrent accesses to WSN data from multiple clients are possible. These clients can implement different applications that process and present the same data at different levels. This capability is extended worldwide when using the Internet, always taking into account the possibility of packet loss and delays introduced by networks.
- This technology opens up new possibilities when we deploy applications that run on mobile devices such as PDAs, SmartPhones, etc., since their wireless interface can be used for communication between both layers.

This paper is divided into the following parts. In Section II we describe in detail the raised architecture. In Section III we show a Matlab example of how data could be collected from a WSN with the raised architecture in a totally transparent way. In section IV we show two examples of how the Physical layer would be implemented in a Bluetooth or a ZigBee WSN. And finally, we expound the conclusions in Section V.
connect the anchor nodes to a Host. The Host should capture in WiFi, UWB, GPS, etc.), one or another interface is used to the technology used to implement it (i.e. Bluetooth, ZigBee, standard manner that the Proxy
physically and for formatting the gathered parameters in a
proxy layer the WSN technology and how the RSS measurements are obtained from it. Now this layer can be changed by another that access to different WSN hardware technology, without any problem. To this end we should simply implement a small program, called port, for each technology chosen to implement a WSN. In Section IV we will show two example implementations of this layer for the cases of Bluetooth and ZigBee technologies.

The main purpose of this layer is to totally abstract the Proxy layer the WSN technology and how the RSS measurements are obtained from it. Now this layer can be changed by another that access to different WSN hardware technology, without any problem. To this end we should simply implement a small program, called port, for each technology chosen to implement a WSN. In Section IV we will show two example implementations of this layer for the cases of Bluetooth and ZigBee technologies.

The main purpose of this layer is to totally abstract the Proxy layer the WSN technology and how the RSS measurements are obtained from it. Now this layer can be changed by another that access to different WSN hardware technology, without any problem. To this end we should simply implement a small program, called port, for each technology chosen to implement a WSN. In Section IV we will show two example implementations of this layer for the cases of Bluetooth and ZigBee technologies.

The main purpose of this layer is to totally abstract the Proxy layer the WSN technology and how the RSS measurements are obtained from it. Now this layer can be changed by another that access to different WSN hardware technology, without any problem. To this end we should simply implement a small program, called port, for each technology chosen to implement a WSN. In Section IV we will show two example implementations of this layer for the cases of Bluetooth and ZigBee technologies.

The main purpose of this layer is to totally abstract the Proxy layer the WSN technology and how the RSS measurements are obtained from it. Now this layer can be changed by another that access to different WSN hardware technology, without any problem. To this end we should simply implement a small program, called port, for each technology chosen to implement a WSN. In Section IV we will show two example implementations of this layer for the cases of Bluetooth and ZigBee technologies.

The main purpose of this layer is to totally abstract the Proxy layer the WSN technology and how the RSS measurements are obtained from it. Now this layer can be changed by another that access to different WSN hardware technology, without any problem. To this end we should simply implement a small program, called port, for each technology chosen to implement a WSN. In Section IV we will show two example implementations of this layer for the cases of Bluetooth and ZigBee technologies.

The main purpose of this layer is to totally abstract the Proxy layer the WSN technology and how the RSS measurements are obtained from it. Now this layer can be changed by another that access to different WSN hardware technology, without any problem. To this end we should simply implement a small program, called port, for each technology chosen to implement a WSN. In Section IV we will show two example implementations of this layer for the cases of Bluetooth and ZigBee technologies.
- **Clean RSSI buffer by ReceiverID**: It cleans the measurement buffer of a given anchor.
- **Start the Physical layer**: It starts the Physical layer by launching the process which is responsible for communicating at low-level with the WSN.
- **Stop the Physical layer**: It stops the Physical layer by sending a `sigint` [7] signal to the process responsible for communicating with the WSN. In this case, all the RSSI buffers stored for the anchor nodes remain intact.

This layer is the most complex layer of the entire architecture because it implements an XML-RPC server which provides the five previous procedures.

Before starting to describe in depth all the parts that make up the Proxy, we must explain the data structures used to store the various RSS and Timestamp buffers for the different mobile devices detected, which are replicated for all the anchors that form the WSN. The Proxy maintains a hash table which is indexed by anchor node. For each entry in this table it hangs a list of the mobile devices that were detected by the anchor. And in turn, for each discovered device it must handle other 2 lists, which will store (Timestamp, RSSI) tuples of all observed measurements.

Figure 3 shows a simplified example of this data structure, where data are shown only for the first 2 anchors of a WSN and in turn only for the first 2 mobile devices detected by one of them. These lists will grow automatically as new devices are detected or as new measurements of existing devices are observed, up to a RSS buffer limit (defined by a variable, `maxData`, in the code). If this limit is exceeded, new measurements will be stored according to a chronologically ordered FIFO queue, where new elements are added at the end and the oldest ones are removed from the beginning. This way, we always have available the last `maxData` measurements for each detected mobile node.

Figure 4 shows a UML class diagram, which represents the classes that form the Proxy layer as well as the relationships and dependencies between them.

We have implemented this layer by using the Apache XML-RPC [3] server implementation due to its simplicity of use, open-source code, excellent documentation and its implementation in Java, a powerful object-oriented language.

While the Physical layer is normally implemented using a low-level language (such as C), we used Java to implement the Proxy layer instead. As said before, there is no restriction in choosing different programming languages to implement different layers.

The signature of the procedures that can be remotely requested to the Proxy, as well as their parameters, their data type and type of the returned data, are
defined in the interface Facade. This facade represents the XML-RPC server requestProcessor that basically defines the signatures of the procedures published by the class MyXmlRpcServlet. This class is really simple because it inherits almost full functionality of the class org.apache.xmlrpc.XmlRpcServlet, which actually implements the XML-RPC server. It is already implemented by the Apache team. To start this server (Proxy layer), we should just run the class ServletServer from the command-line, passing as parameters the path to the port of the Physical and other configuration arguments such as the anchor node identifiers or the UART/USB used ports. This class creates a servlet of type MyXmlRpcServlet and passes it these arguments. Then, it instantiates a Proxy object (pObject) which is set within the Facade by the method init. Any remote request will be answered by the servlet, which will directly delegate into this pObject. For this reason the interface Proxy has the same methods as the interface Facade (except for init). An additional usefulness of this delegation is the capability to rewrite the XML-RPC procedure parameters, if needed, to other most appropriate for the programming language used. We decided to use threads to implement the interface Proxy. This is why ProxyImpl implements Runnable. When the procedure start() is remotely invoked, a new thread is created, which launches the port at physical level. Then, it remains in an infinite loop, continuously parsing everything the port writes to the stdout. As soon as new measurements are detected, the thread updates the device hash table (inside ProxyImpl) and the Timestamp and RSSI tables of the Device objects which hang from each anchor of the hash table.

The stop() procedure posts a sigint [7] signal to the port and the termination of the thread.

The getRSSIs, getRSSIsT and cleanRSSIs procedures have an anchor parameter that represents the anchor identifier. They use this identifier to access the device hash table, to obtain or clean the corresponding measurement list. The getRSSIsT procedure is very similar to getRSSIs but it averages by TransmitterID the data from a list and filters it after a certain timestamp reference instant, before returning it.

II-C. Wrapper

It is implemented in the client side above the Location layer. It implements a XML-RPC client and abstracts to the Location layer all XML-RPC request logic. It can be implemented using a different language than all other layers, but matching the language of the XML-RPC client library. It also abstracts to the higher layer the way of collecting data that is received in streaming mode. This greatly eases its adaptation to process incoming real data (that comes in real time) from the WSN.

II-D. Location

It is responsible for implementing the location algorithm. It feeds on real remote data obtained by the Wrapper in a completely transparent manner.

III. ACCESSING THE WSN FROM MATLAB

The main advantages of having a Wrapper implemented in Java, is that we can directly use it from a high-level simulation environment as Matlab. This is possible because of the intrinsic connection between Matlab and Java virtual machine (JavaVM), where Java objects can be created transparently, and data type conversion between Matlab and Java is automatically handled.

To define a XML-RPC Java client using the libraries provided by Apache, we should simply create a XmlRpcClient object, call its setConfig method with our desired configuration (server URL, userName, password, timeouts, etc.), and finally pass this client as a parameter to the ClientFactory to directly get an instance of our Facade. The local procedure calls to this new facade object will take place on the Proxy server in a transparent manner. However, it is also possible to use a shorter way to invoke these procedures by directly using the execute method of the client object. We have used this second method in our example.

Usually, location algorithms are implemented in Matlab and are proven using simulated data. However, as soon as algorithms work, it is desirable to use them with real RSS data that is collected in real-time from a WSN. By using our small and simple Matlab files shown in Figure 5, which is a complete Java Wrapper implementation, we provide remote access to real RSS data in a totally transparent manner. All the configuration parameters should be first defined in the xmlrpc.CONNECT.m file before running other parts of code.

IV. PHYSICAL LAYER IMPLEMENTATIONS

When we implement a WSN using Bluetooth technology, the network and its operation would look like in Figure 6. Each anchor will be connected to the host by cable and get RSSI data by means of the standardized Bluetooth Inquiry [11][13] procedure. The Physical layer port could easily be implemented maintaining 2 processes, one in an infinite loop sending Inquiry (0x01)(0x0001) HCI commands [11], while the other one parses Inquiry Result with RSSI (0x22) HCI events [11]. For each parsed event, it should write to stdout a Timestamp (the current system time), TransmitterID (mobile node MAC), ReceiverID (anchor MAC) and RSSI.

In the case of ZigBee technology, the WSN and its operation would be similar to Figure 7. A coordinator node [12] of the network could act as a gateway forwarding every packet it receives, directly to a Host. RSS collection is now completely wireless, and any individual anchor would route any packet to the Gateway (0). Any mobile node would periodically broadcast a small packet that would be received by the anchors and the Gateway, and then, the anchors would send detected RSS data to the Gateway. It is necessary, therefore, to implement a small packet relay logic at the anchors and a simple port in the Host, which parses and writes the received packets to the stdout as: Timestamp (the current system time), TransmitterID (mobile node address), ReceiverID (anchor address) and RSSI (packaged into the relayed packets).
V. CONCLUSIONS

This paper introduces a multi-layer architecture that abstracts hardware technology and location algorithms used by a location system based on WSN. The raised architecture is divided into four layers. The Physical layer accesses the WSN at low-level and abstracts the WSN hardware to other layers. The Location layer implements a location algorithm. And finally, two other layers (Proxy and Wrapper) abstract the communication between the previous two layers using the XML-RPC standard.

The main benefits achieved are the reuse and independence of the layers, the use of different languages to implement them, and the possible remotely access to a WSN from multiple clients. The paper also presents a Matlab implementation that shows how to transparently access to real RSS data, gathered from a remote WSN. Also it is shown two possible ways of implementing the physical layer with two different WSN technologies, such as Bluetooth or ZigBee.

VI. ACKNOWLEDGEMENTS

This work has been supported by: 07TIC019105PR (Xunta de Galicia, Spain) and TSI-020301-2008-2 (Ministerio de Industria, Turismo y Comercio, Spain).

REFERENCES