FP7-2010-NMP-ENV-ENERGY-ICT-EeB

TIBUCON

Self Powered Wireless Sensor Network for HVAC System Energy Improvement – Towards Integral Building Connectivity

Instrument:	Small or medium-scale focused research project - STREP
Thematic Priority:	EeB.ICT.2010.10-2 – ICT for energy-efficient buildings and spaces of public use

D2.3 High Level Data Models and Message Structures

Due date of deliverable:	30.09.2011
Actual submission date:	21.10.2011

Start date of project: 01.09.2010 Duration: 36 months

Organization name of lead contractor for this deliverable:	TEKNIKER
Dissemination level:	PU
Revision	Final

Change log

Version	Date	Change
0.1	27.09.2011	Structure of the Deliverable [TEKNIKER]
0.2.	30.09.2011	Deliverable sent to PC.
0,3	14.10.2011	Comments sent to TEKNIKER by PC.
1.0	21.10.2011	Deliverable sent to EC.

Main authors:

	Jorge Berzosa - TEK	Jberzosa@tekniker.es
	Jon Mabe - TEK	jmabe@tekniker.es

Executive Summary [TEKNIKER]

This document deals with the description of the proposed data model and standards for each layer of the TIBUCON project. How these standards fit into the project or which problems must be overcome and the way to solve them are described in the current document,.

Abbreviations

COAP	Constrained Application Protocol
D	Deliverable
DPWS	Devices Profile for Web Services
e.g.	exempli gratia = for example
EC	European Commission
etc.	et cetera
EXI	Efficient XML Interchange
HTTP	Hypertext transfer Protocol
HVAC	Heating, Ventilation, Air Conditioning
i.e.	Id est = that is
MAC	Medium Access Control
MTU	Maximum transfer Unit
OGC	Open Geospatial Consortium
SP-MM-WSN	Self Powered Multi Magnitude Wireless Sensor Networks
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
SWE	Sensor Web Enablement
TIBUCON	Self Powered Wireless Sensor Network for HVAC System Energy Improvement - Towards Integral Building Connectivity
WP	Work Package
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WT	Work Task
XML	Extensible Markup Language

Contents

Figures

1 Introduction

Even though the lowest communication layer (MAC layer) has been specifically designed for TIBUCON, it is desirable for the upper layers to be implemented by mean of standards. This is a key objective in order to achieve interoperability and service discovery.

The intermediate layers of the WSN protocol stack are the ones where the commercial and standard solutions are oriented. Some of those solutions are overlapped with lower and higher layers in order to achieve a certain functionality. Those standards tended to be designed for rich resource environments and devices. However, in the last years a lot of effort has been made to adopt then to systems with limited capabilities, such as TIBUCON sensor nodes. In these nodes not only the computational resources are constrained but also the power consumption must be keep to the very minimum.

The upper layers in WSN are those intended for achieving a standard interface for general applications, therefore, these are the layers where the standardization effort must be put. This mentioned standardization should be understood in two different steps: first the accessibility and second the understanding of the information. For TIBUCON project the accessibility of the information will be achieved through the inclusion of the 6LowPAN standard which empower the WSN nodes with an approximation to IP connectivity. In addition, Service Oriented Architectures will be mentioned as the facto standard to exchange information and services in a distributed cooperative scenario.

On the other hand, the interpretation and understanding of the gathered information will be based on standard data models formatted in a well-known fashion.

2 Selected Communication Standards

Interoperability is one of the leading factors when choosing a wireless protocol. For this reason standards will be selected wherever possible. However, the system has very high constraint regarding resources (specially available energy) and is the main concern of the project.

The architecture of the system is the one depicted in the Figure 2-1.

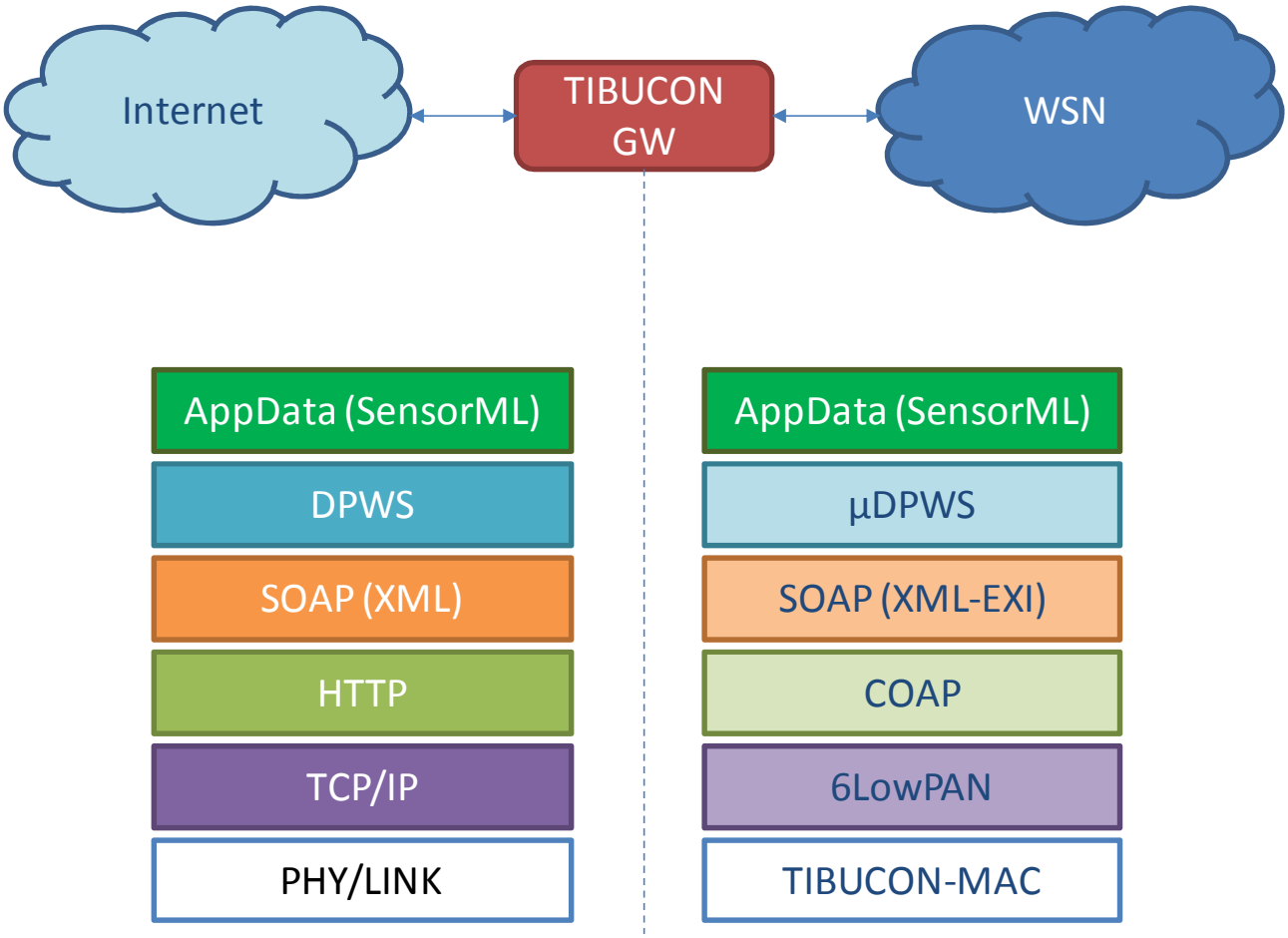

Figure 2-1: TIBUCON architecture

The Mac layer is the responsible of the management of the network and packet routing. It is specially designed for ultra low consumption so besides the common limitations associated to WSNs (packet size, computation capabilities, etc.), it has stronger restrictions like very low duty cycle or poor sensor connectivity.

The standards used in the upper layers has been chosen due to their interoperability and functionality. The final aim is to allow easy service/resource discovery and consumption from clients outside the WSN and hosted in different platforms.

2.1 6LowPAN

6LowPAN[1] allows the routing of ipv6 packets from/to the WSN. WSNs are connected to the network via gateways that act as interfaces and often translate the external protocols/data to the inner network and vice versa. In order to achieve interoperability, 6lowPan is the basic pillar that will allow high layer protocols to be used across routers without the necessity of an explicit firmware.

6LowPAN defines the frame format for transmission of IPv6 packets and the binding of IPv6 and local addresses to be used over IEEE 802.15.4 networks. Actually, 6LowPan could be understood as a compressed IPv6 format. Although, it is designed to work over IEEE 802.15.4 constraint it is not strongly dependant of the MAC layer or the routing protocol. This means that it can be implemented over different MAC protocols with little effort.

The MTU size of IPv6 packets is much larger than the one supported in IEEE 802.15.4 (127 octets at most). 6LowPAN defines a fragmentation and reassembly adaptation layer to deal with this issue. This is an important feature because in the system defined by TIBUCON, large data streams need to be transmitted in certain situations (e.g. service discovery/identification procedure).

2.2 RESTful Constrained Application Protocol (CoAP)

CoAP is a substitute of HTTP for constrained environments described in the internet draft [2]. The main features are:

- RESTful protocol design minimizing the complexity of mapping with HTTP.
- Low header overhead and parsing complexity.
- URI and Content-type support.
- Support for the discovery of resources provided by known CoAP end-points.
- Simple subscription for a resource and a resulting notification mechanism.
- Simple caching based on max-age.

TIBUCON supports a partial implementation of the CoAP protocol as some of the features are fulfilled by means of higher layers. In fact, it is used just as a simple and more efficient encoding of HTTP.

2.3 Simple Object Access Protocol (SOAP)

SOAP[3] provides a messaging framework for the definition of web services. SOAP is based on XML and is widely used over the network. However, SOAP imposes the strongest challenge regarding WSN: the transmission of big bulks of data in XML format. With the use of compressors (such as EXI) the data size decreases notably although in some cases it remains higher than supported.

2.3.1 XML-EXI

As showed in the internet draft[4] EXI[5] is the most promising compressor. The EXI format uses an algorithm based in information and formal language theories and practical techniques for efficient encoding of XML formatted data.

The most interest mode of operation of EXI is named as schema-informed mode. In this mode a known schema is used to form a specific grammar and state-machine increasing notably the compression rate. On the other hand, the simplified state machine allows the implementation of EXI in constrained devices avoiding the use of a full XML parser.

However, no matter how effective EXI is, there are XML streams with a size of thousands of bytes. In this case some other techniques or workarounds must be depicted to allow the sending of this streams in just a few packets or minimizing the need repeat the transmission.

2.3.2 TIBUCON approach

The major (inevitable) drawback of EXI is that the first occurrence of each string cannot be compressed. Each string will use valuable space and in a common SOAP/XML scenario the amount of different strings is normally high.

The most efficient mode of EXI is the schema-informed mode. In the other modes is not schema-informed the first occurrence of each label cannot be encoded (similar to strings). However the gateway cannot store the schema of all possible SOAP based standards.

To avoid these overload four mechanisms has been designed:

- Schemas: a first approach is to store the schemas that are known to be used (in our case DPWS/ SensorML schemas). However to assure a more wide and efficient solution each node stores its own schemas and send them to the gateway under demand. To avoid unnecessary retransmission each schema should be labeled with an unique identifier.

- Local Translation Table (LTT): each node stores a table with a value pair of the type (string, Id) and sends it to the gateway under demand. This table is specific of each node and should be used to encode strings in a more compact format.
- Global Translation Entities (GTEs): each node stores a set of strings labeled with an unique identifier and sends it to the gateway on demand. Note that with this mechanism not just simple string can be encoded but also whole text chunks of a XML file.
- Templates: each node stores a set of XML templates labeled with an unique identifier and sends them to the gateway on demand. A XML template is composed of two elements: the XML text itself containing the static data and an array of offsets describing the position of the dynamic data. Templates can have a global or local scope.

When a node connects to a network for the first time, the gateway asks for its LTT and stores it. Then the node will be asked for the schemas it uses. If the gateway does not have those schemas already, it will request them. Finally, the list of unknown GTEs and templates will be recovered.

Although the Schemas and GTEs will be compressed they will require many transmission packets to be delivered. Thanks to the labeled approach this will only happen once per network. It is worth noting that, with the exception of LTTs, the listed mechanisms offer recurrent functionality.

Optionally if the scope in which the gateway is going to be deployed I known a priory, the schemas and GTEs can be downloaded via internet to speed up the commissioning of the deployment.

EXI protocol must be modified to integrate these mechanisms into the encode/decode process.

2.4 Devices Profiles for Web Services (DPWS)

DPWS[6] defines a minimal set of implementation constraints to enable secure Web Service messaging, discovery, description and eventing on resource-constrained devices.

The main purpose of DPWS is to bring web services to small embedded devices. That way, your devices can communicate with each other or other DPWS enabled devices and applications using and standardized protocol. TIBUCON adds a restriction to this scope in order to reduce the traffic load: the nodes within the WSN network are not allowed to communicate directly between them, i.e. any communication must go through the gateway.

DPWS has many appealing characteristics such as discovery/description and eventing (which avoid the use of polling procedures). However, the collaboration of the gateway is necessary in order to minimize the traffic load of the network. For instance, it can store the discovery and description packets of each connected node to avoid unnecessary message traffic.

uDPWS[7] is an implementation of the Devices Profile for Web Services (DPWS), especially for memory-efficient networked embedded systems and wireless sensor networks. It is designed to work on microcontrollers with small amounts of memory.

DPWS adds even more load to the communication packet size and message traffic. Apart from the mechanisms described in the previous chapter, TIBUCON uses some advices from the internet draft [8]:

- A Device must include the transport specific addresses in its *Hello and Probe Match* messages.
- The fields for the transport specific addresses are mandatory instead of optional to avoid Resolve messages.
- A Device should include all device types in *Hello and Probe Match* messages.
- All SOAP-over-UDP messages inside the 6LoWPAN network must use the port 61616 as target port. (Exact port to be defined).
- Devices inside the 6LoWPAN network must listen to the IPv6 multicast address: FF02::C0. (Exact address to be defined).
- Clients inside the 6LoWPAN network must listen to the IPv6 multicast address: FF02::C1. (Exact address to be defined).
- DPWS defines one IPv4 and one IPv6 multicast addresses to be used for discovery message exchange. But DPWS differentiates between device and client roles. Hence there are messages to be processed by clients (*Hello*) and by devices (*Probe*, *Resolve*) exclusively. Usage of one and the same address for all these messages implies overhead in sending to and receiving of these messages independent of the nodes role. Inside 6LoWPAN networks, different addresses have to be used. The mapping into compliant addresses is done by the edge router of the 6LoWPAN network.
- In managed mode, a device and service registry is applied. In managed mode, one discovery proxy should be deployed at the edge router to hide expensive external multicast messages from the 6LoWPAN network and omit multicast flooding.

2.5 Communication Message Structures

The simplified message structure is shown in the Figure 2-2. Each colored section represent the header of the associated layer protocol.

Figure 2-2: TIBUCON message structure

The last section is codified in XML-EXI format and is composed of all the upper layer protocols (SOAP, DPWS). This protocols are implemented within the XML format and can not be easily decomposed in detail in their different sections. SOAP is based on XML and defines the necessary mechanisms for the execution of remote functions and transmission of data. DPWS is built on top of SOAP and uses the provided mechanisms to implements its own functionalities. In some sense SOAP envelopes DPWS which in turn envelopes the application data.

3 Data Models

3.1 SensorML

OGC Sensor Web Enablement (SWE)[9] standards enable developers to make all types of sensors, transducers, and sensor data repositories discoverable, accessible, and useable via the Web. Standards make it possible for users to assemble sensor systems and components together efficiently, protect investments, reduce likelihood of dead-end technologies, products or approaches and allow for future expansion and encourage collaboration between Sensor Web components and users.

Sensor Web Enablement allows for the integration and analysis of streams of sensor data from multiple and diverse sensors in a standards-based and thus interoperable manner. For instance, observations from water quality sensors can be fused with those from weather instruments and satellite remote sensing instruments. Collection, management and analysis of these data can be automated and adaptive, handling the disruption of service from some sensors or the addition of new sensors.

SensorML[10] is a component of the OGC SWE framework. SensorML provides a framework for the definition of the parameters and all related concepts of sensors and sensor systems.

The purposes of SensorML are to:

- Provide descriptions of sensors and sensor systems for inventory management.
- Provide sensor and process information in support of resource and observation discovery.
- Support the processing and analysis of the sensor observations.
- Support the geolocation of observed values (measured data).
- Provide performance characteristics (e.g., accuracy, threshold, etc.).
- Provide an explicit description of the process by which an observation was obtained (i.e., it's lineage).
- Provide an executable process chain for deriving new data products on demand (i.e., derivable observation).
- Archive fundamental properties and assumptions regarding sensor systems.

One of the advantages of SensorML is that it does not depend upon the presence of the other SWE components. This is an interesting feature because a sensor or sensor reading in

SensorML format is self-describing so it can be used within any system and be shared easily and transparently with interested applications.

3.1.1 SensorML in TIBUCON

SensorML is strongly based on XML encoding. This adds even further load to the message size, increasing the bandwidth needs.

In order to avoid this overhead, SensorML formatted data will use Global Templates extensively. To the known scope of TIBUCON is easy to define a set of predefined Global Templates for shared data types (temperature, humidity,…).

In Annex-A various structures of SensorML data types for TIBUCON are described. The specific structure will depend in the sensors each node has.

4 Conclusions

Is this document the architecture and data model of the project TIBUCON has been described. The standard selected for each layer and its contribution to the project have been depicted.

Although there are some problems regarding the adaptation of the high layer standards to a WSN system, some solutions have been proposed in order to overcome them. If this measures probe to be efficient the main objective of TIBUCON in this scope will be fulfilled: a system with capabilities of service/resource discovery based on standards available over an ultra-low consumption WSN.

ANNEX-A. TIBUCON SensorML Data Types Examples

Some text has been omitted (specially headers) for simplicity.

Node with a single temperature sensor:

```
<swe:field name="Temperature">
    <swe:Quantity definition="urn:TIB:Temperature">
        <swe:uom code="Cel"/>
        <swe:constraint>
            <swe:AllowedValue id="TemperatureRange">
                <swe:interval>-20.0 60.0 </swe:interval>
            </swe:AllowedValue >
        </swe:constraint>
        <swe:value> 23.0 </swe:value>
    </swe:Quantity>
</swe:field>
```

Node with a temperature and humidity sensor:

```
<swe:DataRecord>
    <swe:field name="Temperature">
        <swe:Quantity definition="urn:TIB:Temperature">
            <swe:uom code="Cel"/>
            <swe:constraint>
                <swe:AllowedValue id="TemperatureRange">
                    <swe:interval>-20.0 60.0 </swe:interval>
                </swe:AllowedValue >
            </swe:constraint>
            <swe:value> 21.3 </swe:value>
        </swe:Quantity>
    </swe:field>
    <swe:field name="RelativeHumidity">
        <swe:Quantity definition="urn:TIB:RelativeHumidity">
            <swe:uom code="%"/>
            <swe:constraint>
                <swe:AllowedValue id="RelativeHumidityRange">
                    <swe:interval>0.0 100.0 </swe:interval>
                </swe:AllowedValue >
            </swe:constraint>
            <swe:value> 53.0 </swe:value>
        </swe:Quantity>
    </swe:field>
</swe:DataRecord>
```

Node with a temperature sensor and remaining battery monitoring:

```
<swe:DataRecord>
    <swe:field name="Temperature">
        <swe:Quantity definition="urn:TIB:Temperature">
            <swe:uom code="Cel"/>
            <swe:constraint>
                <swe:AllowedValue id="TemperatureRange">
                    <swe:interval>-20.0 60.0 </swe:interval>
                </swe:AllowedValue >
            </swe:constraint>
            <swe:value> 21.3 </swe:value>
        </swe:Quantity>
    </swe:field>
    <swe:field name="RemainigEnergy">
        <swe:Quantity definition="urn:TIB:RemainigEnergy">
            <swe:uom code="C"/>
            <swe:value> 200.0 </swe:value>
        </swe:Quantity>
    </swe:field>
</swe:DataRecord>
```

References

[1] 6LoWPAN, The wireless Embedded Internet, by Zach Shelby and Carsten Bormann.

[2] http://tools.ietf.org/html/draft-shelby-core-coap-01, accessed Sept 30, 2011

[3] http://www.w3.org/TR/soap/, accessed Sept 30, 2011

[4] http://tools.ietf.org/html/draft-shelby-6lowapp-encoding-00, accessed Sept 30, 2011

[5] http://www.w3.org/XML/EXI/, accessed Sept 30, 2011

[6] Introduction to DPWS, http://www.pialek.eu/blog/programming/dpws/16-introduction-to-dpws-devices-profile-for-web-services.html, accessed Jan 13, 2011

[7] WS4D-uDPWS - The Devices Profile for Web Services (DPWS) for highly resource-constrained devices, http://code.google.com/p/udpws/, accessed Sept 30, 2011.

[8] http://tools.ietf.org/html/draft-moritz-6lowapp-dpws-enhancements-01, accessed Sept 30, 2011

[9] OGC Sensor Web Enablement, http://www.opengeospatial.org/projects/groups/sensorweb, accessed Sept 30, 2011

[10] http://www.opengeospatial.org/standards/sensorml, accessed Sept 30, 2011