BEMO-COFRA

Brazil-Europe Monitoring and Control Framework

(Project No. 288133)

D4.3 WSAN network management

Published by the BEMO-COFRA Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme
and
Conselho Nacional de Desenvolvimento Científico e Tecnológico
Objective ICT-2011-EU-Brazil

Document version: 0.5 Submission date:

Document control page

Document file:	BEMO-COFRA D4.3 WSAN network management.docx
Document version:	0.3
Document owner:	Markus Taumberger (VTT)
Work package:	WP4 – Large Scale Wireless Sensor and Actuator Networks
Task:	T4.3 – Support for WSAN network management
Deliverable type:	P
Document status:	☒ approved by the document owner for internal review ☒ approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of Changes made
0.1	Markus Taumberger, Mikko Ala-Louko	2012-02-24	Initial version
0.2	Markus Taumberger, Mikko Ala-Louko	2012-03-02	Related work and terminology added
0.3	Markus Taumberger, Janne Raappana	2013-09-19	Implementation section updated
0.4	Hussein Khaleel	2013-10-30	Implementation architecture updated
1.0	Markus Taumberger		Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Ferry Pramudianto	2013-10-31	**Accepted with minor comments**

Index:

1. Executive summary 4
2. Introduction 5
2.1 Purpose, context and scope of this deliverable 5
2.2 Background 5
2.3 Scope of the work 5
3. Related work 6
3.1 Relevant Standards 6
3.1.1 FCAPS Model for Network Management 6
3.1.2 Simple Network Management Protocol (SNMP) 6
3.1.3 Abstract Syntax Notation One (ASN.1) 6
3.2 State-of-the-art 6
3.2.1 OBELIX Framework 6
3.2.2 MotePlat 6
3.2.3 Monitoring Distributed Systems 7
3.2.4 Monitoring Wireless Sensor Networks 7
4. Requirements 8
4.1 Domain-independent requirements 8
4.2 Requirements from the manufacturing industry 8
5. Specification 9
5.1 Terminology 9
5.2 Features 9
5.3 Overall architecture 9
5.3.1 Physical network 10
5.3.2 Overlay projection 10
5.3.3 Administration overlay network 10
6. Implementation 11
6.1 Architecture 11
6.1.1 SNMP agent 11
6.1.2 Managed WSAN information and parameters 13
6.1.3 SNMP Manager 13
6.1.4 WSAN Proxy 13
References 14

1. Executive summary

This deliverable describes the mechanisms designed and implemented for network management of the wireless sensor and actuator (WSAN) network in the BEMO-COFRA platform. The mechanisms allow the monitoring and control of the network, i.e. status data like link quality and used channels can be observed and nodes' parameters and actuators can be changed.

The developed protocols are used in Task 5.2 by the monitoring and control tool that acts as user interface for the network management.

2. Introduction

2.1 Purpose, context and scope of this deliverable

2.2 Background

Wireless Sensor and Actuator Networks (WSAN) are becoming more and more part of everyday lives and also start finding their way into the manufacturing industry. The growing amount of heterogeneous network nodes and their spatial distribution over potentially big areas make remote administration of these distributed systems necessary. It is laborious to specify and implement this infrastructure from scratch for every platform; there is a need for one multi-platform monitoring infrastructure for heterogeneous networks. Typically the network nodes have constraint resources regarding computational power, memory size and are often battery-powered, which is why the solution has to be lightweight, especially concerning the network load.

Administration is required during setup and operation of the WSAN, both phases have different requirements. In the setup of a WSAN it is important to assign parameters to individual nodes, such as name or location, while during operation parameters for the whole network might be tuned and the state of the network needs to be monitored to detect error conditions. Also the monitored data needs to be processed and presented as meaningful data to users with accordingly adjusted level of detail, e.g. maintenance staff or production line manager.

Aim of this work is to create a general purpose and re-usable administration framework for WSANs that a monitoring and administration tool (to be developed in Task 5.2 of WP5) can connect to and interface with end-users. Existing approaches for network management and distributed systems have been used as starting point and were extended where needed to fulfil the requirements.

2.3 Scope of the work

In this work we created an administration framework to monitor and manage distributed systems in general and specifically WSANs. Therefore we adopted *Network Management* and *Distributed Systems* approaches to administrate WSANs in the context of resource constraint, large scale and highly spatial distributed deployments. While traditional network management mainly covers administration of the communication links of nodes we focus more on the administration of the actual nodes (sensors and actuators) that are interacting with the environment to ensure system operation.

The designed administration framework consists of administration agents, which monitor and manage the network nodes, and a protocol for exchanging data. It supports *a) safety*, meaning that the monitoring is as non-intrusive as possible guaranteeing that the actual operations of the system are not affected, *b) web-based* remote monitoring, *c) dynamic data aggregation* at network level, and *d) scaling* for large scale deployments.

3. Related work

3.1 Relevant Standards

3.1.1 FCAPS Model for Network Management

The ISO FCAPS model for network management has been developed for the information technology management of organisations (FCAPS 1998). The acronym stands for *fault*, *configuration*, *administration*, *performance*, *security* and the model specifies management processes and goals for all these elements in a network-centric approach.

We ensured a general purpose approach by using this standard as guideline of our work covering *a) fault management*, to recognise, isolate, collect and log network faults, *b) configuration management*, where device configurations are gathered, set and tracked, *c) administration management*, that is used to gather usage statistics, *d) performance management*, to determine the system efficiency, and *e) security management*, that controls the data access in the network.

3.1.2 Simple Network Management Protocol (SNMP)

The Simple Network Management Protocol is the standard protocol for managing devices connected via an Internet Protocol (IP) network (SNMPv3 2004). The SNMP agent is sending notifications autonomously and responses on request to the manager, optionally with cryptographic security. The data is exchanged in the format of protocol data units (PDUs). The extensible design of SNMP allows the specification of the actual management data by using object identifiers (OID) that are defined in management information bases (MIBs). OID variables can be set or read through SNMP.

In our framework the administration agents communicate via SNMP-compliant messages. The widespread implementation of TCP/IP protocols on various physical network standards allows the seamless integration of the administration framework on many platforms. The static assignment of management entities is replaced with a dynamic multi-level management hierarchy.

3.1.3 Abstract Syntax Notation One (ASN.1)

The standard of Abstract Syntax Notation One specifies a universal notation of data in computer systems (ASN.1 2008). It allows a platform-independent representation of data that can be stored, exchanged, processed and validated within different systems. Different information encodings are specified that range from XML to binary.

The data structures of the administration protocol are specified using the ASN.1 notation. Also the data encoding for communication follows the ASN.1 standard.

3.2 State-of-the-art

3.2.1 OBELIX Framework

OBELIX is a framework for the network monitoring of distributed wireless sensor networks (Riggio et al 2011). It uses a distributed architecture for monitoring wireless multi-hop networks and is able to support runtime changes in the operations performed by different nodes. The framework defines a set of algorithms and protocols able to ensure robustness of the monitoring infrastructure with respect to nodes failure and to changes in the network topology. A prototype of the proposed architecture and protocols has been implemented and validated in a real-world testbed of small scale.

While this approach is also optimised for low traffic, multi-hop wireless networks and recovery from node failures, our solution is more lightweight and especially supports resource constraint sensors, improves the scalability and eases up integration into heterogeneous networks.

3.2.2 MotePlat

Moteplat is a monitoring and control platform for WSNs programmed in TinyOS (Sun et al 2006). It supports one sink per sensor network and connects all sinks with a management server that forwards monitoring data to and receives control commands from the user(s). Sensor and sink nodes as well as the management server are statically assigned.

We created a fully distributed design where the roles of nodes are not static, but adopt according the current state of the network, i.e. amount and types of nodes. No centralized server (PC) is needed for operation.

3.2.3 Monitoring Distributed Systems

The work of (M.-S. et al 1993) describes a functional model for monitoring distributed systems. Separate management interfaces are used in the network nodes for monitoring. The monitoring process is divided into the activities Generation, Processing, Dissemination and Presentation. The level of abstraction increases with each step that the monitored data passes, so that monitoring reports are converted into monitoring traces. Validation of monitoring information is carried out to ensure correct monitoring.

Wireless Sensor and Actuator Networks are distributed systems and the principles for their administration are similar. We adapted basic ideas, like monitoring reports, and optimised them to fit the criteria of resource constraint nodes with wireless communication.

3.2.4 Monitoring Wireless Sensor Networks

This monitoring infrastructure can indicate system failures and resource depletion in Wireless Sensor Networks (Zhao et al, 2003). The architecture offers three mechanisms to compute aggregates (dump, scan and digest) that allow following the state of the sensor network without centralised collection of monitored data because of the need of energy-efficient communication. Architecture for sensor network monitoring, but main focus on efficient data processing with three levels of monitoring: dump, scans, digests.

4. Requirements

4.1 Domain-independent requirements

Non-functional

- *Ubiquitous administration*. The administration can be carried out by connecting either physically or remotely to any node of the WSAN. That node becomes the gateway for interfacing with the user and the hierarchy of the monitoring and management infrastructure will automatically adopt accordingly.
- *Low-overhead monitoring*. The administration framework does not interfere with the normal operations of the WSAN.
- *Adjustable buffering*. Monitoring data can be sent in real-time or buffered and sent for example when the node communication is invoked by the application.
- *Scalability*. The protocol scales as the network grows by adding / removing manager roles to nodes.
- *Platform independent*. The framework should not depend on any specific hardware / software platform or communication protocol. Integration into different WSAN platforms should be easy.
- *Data aggregation*. The monitored data has to be condensed by processing and interpretation along the management hierarchy in order to reduce communication overhead.
- *General purpose*. The administration functionality can be configured in order to serve any kind of sensor network application. Support for dynamic (during runtime) and static (at compile time) configuration of framework parameters has to be available.
- *Heterogeneity*. Nodes may be different, in terms of energy supply (battery vs. mains), computational power (sensor vs. set-top box) or networking (Zigbee vs. broadband) capabilities. The roles of nodes in the administration framework will be assigned based on a configurable interpretation of those criteria.
- *Robustness*. The framework has to cope with the unpredictable nature of WSANs, it can operate in environments of appearing / disappearing nodes and unreliable communication links.

Functional

- *Data items*. The framework enables logging of data items inside nodes and their local or distributed storage.
- *Data management*. The data items are managed by the framework with data aggregation and access mechanisms.
- *Remote administration*. Data items have to be accessible from any point of the network.

4.2 Requirements from the manufacturing industry

- *Right level of detail*. The framework allows the retrieval of information at am adjustable level of detail to allow providing different users with the information they need for executing their work.
- *Reliability*. The collected data has to be correct, serious errors must be reported promptly and reliably.

5. Specification

5.1 Terminology

We designed a **framework** for the **administration** of distributed systems. Administration means in our context **monitoring**, retrieving data from the devices, and **management**, data processing and modifying parameters, of the network nodes. The framework consists of **administration agents**, software that is running on the network nodes, and a **protocol** for exchanging data.

We refer the actual hardware devices forming the WSAN as **Physical Nodes (PN)** and the software administration agents of the overlay network as **Administration Overlay Nodes (AON)**. Additionally, the **Parent nodes** manage their **child nodes** in the overlay network.

The administration agents access data in the network nodes through **Data Items (DI)** that consist of data such as sensor / actuator / state, type, timestamp and id. Data items are local or global and monitored or managed by the agents. Local DIs are only stored in the respective parent node, while global DIs are replicated across the network for spatial reuse and data preservation. Monitored DIs are read-only, managed DIs can also be modified. The resulting four data item types are:

- **Local Monitored Data Item ($\mathbf{DI}_{\mathbf{Monitored}}^{\mathbf{Local}}$)**
- **Local Managed Data Item ($\mathbf{DI}_{\mathbf{Managed}}^{\mathbf{Local}}$)**
- **Global Monitored Data Item ($\mathbf{DI}_{\mathbf{Monitored}}^{\mathbf{Global}}$)**
- **Global Managed Data Item ($\mathbf{DI}_{\mathbf{Managed}}^{\mathbf{Global}}$)**

5.2 Features

- Decoupling from physical devices and wireless link by building an administration overlay network.
- Simple acknowledgement mechanism to detect disappearing parents / children and to a limited extend lost messages.
- Administrating devices of the network rather than the network itself.
- Each node can dynamically change its role and is either manager and managed object or only managed object.
- "Soft-start" functionality to avoid flooding the network at start-up.
- Support for pushing and polling of data.
- Transmission of data items can be configured according their type (real-time, …) from immediate sending to delayed sending by grouping in a buffer.
- Replication of global monitored / managed data items for spatial reuse and data preservation.
- The administration overlay network is realised as a tree with multi-level hierarchy to ensure scalability, i.e. parents are the managers of their children.
- The protocol scales as the network grows and adds / removes hierarchy layers as needed when the amount of children of a parent reaches a configurable threshold.
- Data exchange protocol resides in the application layer.

5.3 Overall architecture

The physical network is projected onto the administration overlay network as depicted in *Figure 1*. While the physical network can be of any kind (mesh, star, tree), the overlay network is a hierarchical tree which allows the optimization of the administrative communication traffic and the systematic management and monitoring of the physical nodes.

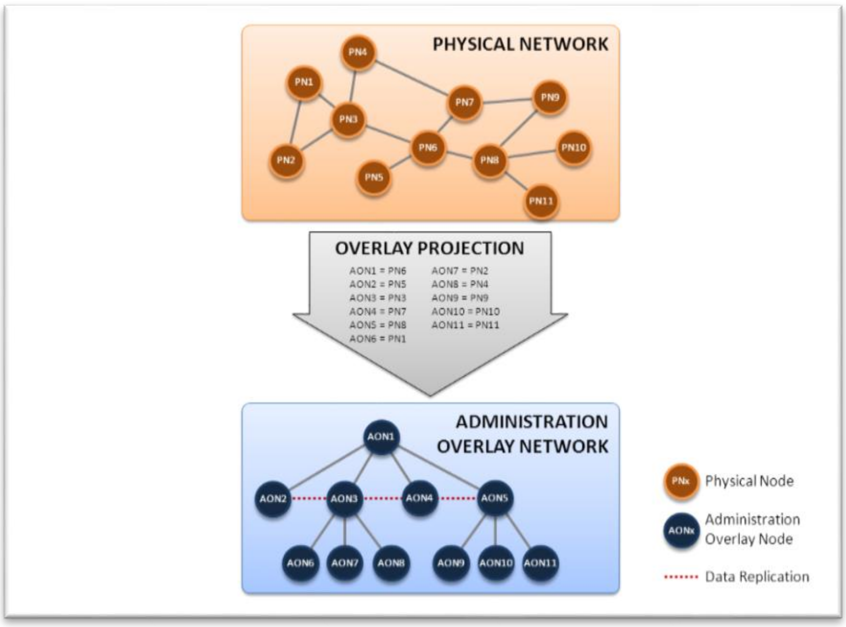

Figure 1: Overall architecture

The projection is dynamic, meaning that the position of the AONs in the overlay network may change over time according to changes in the physical network, for example nodes joining / leaving or disappearing / appearing communication links between nodes. Also during the projection several nodes will be assigned to act as data replicators to ensure data integrity in the case of node failure.

5.3.1 Physical network

- Ad-hoc network (nodes joining / leaving randomly)
- Heterogeneous network devices (sensors, routers, gateways, …) with different computational power, energy supply and amount of memory
- Arbitrary network topology: mesh, star, tree

5.3.2 Overlay projection

- Continuously map the physical devices to the overlay network considering:
    - communication distance
    - amount of children
    - data replication
- "Soft-start" to limit administrative communication overhead at startup

5.3.3 Administration overlay network

- Management of data items
- Data replication
- Collection of monitoring reports

6. Implementation

6.1 Architecture

The network management layout was defined such that it can be integrated into the Multi-radio WSAN architecture, presented in Deliverable 4.2. Consequently, the network management architecture for the considered system is shown in Figure 2.

The data flows between the different multi-radio networks and their corresponding Sink MRNs are implemented with customized protocols. Each Sink MRN features a Data Storage where it stores the incoming data.

The considered system adopts the Simple Network Management Protocol (SNMP) in order to communicate network-related information. For this purpose, each Sink MRN is equipped with an SNMP Agent extension that implements the necessary translation of information into the SNMP format then, whenever required, data are exchanged with the SNMP Manager using standard Traps in the up-link and Requests in the down-link.

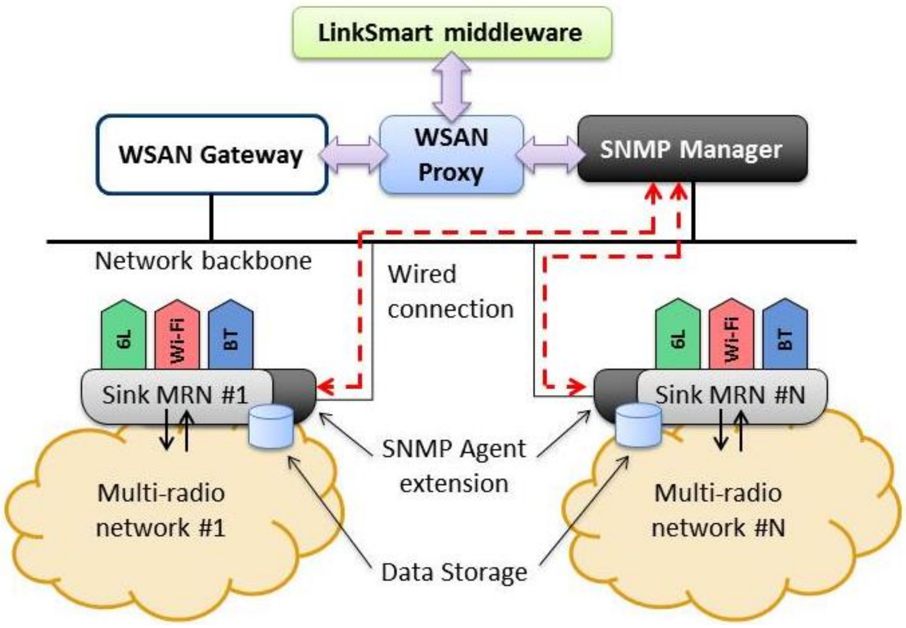

Figure 2: WSAN network management architecture

6.1.1 SNMP agent

In mib-init.c file, all the objects in the MIB-file are defined. It is possible to add more objects to deliver more information to the SnmpManager. At the beginning of the file, all the common prefixes are defined. At function mib_init() at the end of the file, objects are created, their datatype and initial value are defined, object is defined as writable or read-only, and then the assigned function pointers are run whenever objects are accessed through snmpget and snmpset. If function pointers are not assigned for the object, it is just read or writen normally whenever it is accessed.

Example of a “normal” object initialization:

```
if (add_scalar(&oid_sysname, 0, BER_TYPE_OCTET_STRING, "Remote node", 0, 0) == -1) {
        return -1;
}
```

Example of a READ_ONLY object initialization:

```
if (add_scalar(&oid_acceltstamp, FLAG_ACCESS_READONLY, BER_TYPE_TIME_TICKS, 0, 0, 0) == -1)
{
        return -1;
}
```

Example of the initialization of an object which have additional function to be performed whenever accessed:

```
if (add_scalar(&oid_sysuptime, 0, BER_TYPE_TIME_TICKS, 0, &getSystemUpTime,
&setSystemUpTime) == -1) {
        return -1;
}
```

Creating trap-messages:

Calling function snmp_trap() generates trap message and sends it to the border-router. It takes a couple of arguments as a parameter: u8t generic_trap, u8t specific_trap, varbind_list_item_t* vbind_first_ptr. The most important is vbind_first_ptr, which contains a pointer to the first object to be sent in trap-message. Varbind-objects are a linked list which needs to be created before calling the snmp_trap-function. Creating this list is not very simple actually, but you will find an example in PROCESS_THREAD(accelerometer_process, ev_2, data_2) {

First of all, the following variables are created.

```
        static mib_object_t* object;

        static varbind_list_item_t vbind_item_1;

        static varbind_list_item_t vbind_item_2;
```

Each varbind_list_item_t contains bindable object (OID and value) and it happens as follows:

```
        vbind_item_1.varbind.oid_ptr = &oid_accelx;

        object = mib_get(&vbind_item_1.varbind);

        vbind_item_1.varbind = object->varbind;
```

Object-pointer is used as temporary pointer where the bindable object (oid_accelx at this case) is loaded by using mib_get()-funciton. Then we determine initialize second variable bind list item:

```
        vbind_item_2.varbind.oid_ptr = &oid_accely;

        object = mib_get(&vbind_item_2.varbind);

        vbind_item_2.varbind = object->varbind;
```

Finally, when all the bindable items are created, they have to be linked with each other:

```
        vbind_item_1.next_ptr = &vbind_item_2;
```

Now, the snmp_trap()-function can be called:

```
        snmp_trap(6,17, &vbind_item_1);
```

Use Wireshark to see if you have succesfully created and sent the trap-message.

6.1.2 Managed WSAN information and parameters

WSAN nodes provide the following information and capabilities to the Sink MRN:

- Application operational parameters: sensor type, sampling rate, status whether started or stopped.
- Sensing data: clamp states and 3-axis accelerometer readings.

In addition, the Sink MRN has the capability to perform control actions on the WSAN nodes' applications; For instance, the Sink MRN can start, stop, and change the sampling rate of the sensing application.

Using the collected information from a multi-radio network, a Sink MRN calculates packet loss rates and end-to-end delays for the monitoring nodes.

6.1.3 SNMP Manager

This component uses the Java library SNMP4J, its task is to collect information from the SNMP Agents that expose WSAN-related information. The SNMP Manager decodes SNMP-packets and sends the information to the WSAN Proxy. This connection is simple UDP-format and the protocol is user defined.

The SNMP Manager can also send Get/Set Requests, when required, to SNMP Agents in order to control WSAN nodes' parameters.

6.1.4 WSAN Proxy

This is a software module that establishes the connection between the overall multi-radio WSAN system and the LinkSmart middleware. In the context of network management, the WSAN Proxy receives information from the SNMP Manager, and then provides such information to the LinkSmart environment (by publishing events).

The WSAN Proxy also notifies the SNMP Manager in case control actions are required to be performed for WSAN applicatons.

References

(ASN.1 2008)	ISO/IEC 8824 and 8825 (2008), "Information technology – Abstract Syntax Notation One (ASN.1)".
(FCAPS 1998)	ISO/IEC 10040 (1998). "Information technology - Open Systems Interconnection - Systems management overview".
(M.-S. et al 1993)	Mansouri-Samani, M., Sloman, M. (1993). "Monitoring Distributed Systems", IEEE Network Magazine, November 1993 Page(s): 20 – 30.
(Riggio et al 2011)	Riggio, R., Gerola, M., Miorandi, D., Zanardi, A., Jan, F. (2011). "A Distributed Network Monitoring Framework for Wireless Networks", 3rd IFIP/IEEE Workshop on Management of the Future Internet.
(SNMPv3 2004)	IETF (2004). RFC 3411 – RFC 3418.
(Sun et al 2006)	Sun, L., Sun, Y., Shu, J., He, Q. (2006). "MotePlat: A monitoring and control platform for wireless sensor networks", Grid and Cooperative Computing Workshops, 2006. GCCW'06.
(Zhao et al, 2003)	Zhao, J., Govindan, R., Estrin, D. (2003). "Computing Aggregates for Monitoring Wireless Sensor Networks", Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE, 11 May 2003 Page(s): 139 – 148.