European Commission
Information Society and Media

SEVENTH FRAMEWORK PROGRAMME

Large Scale Integrating Project

EXALTED

Expanding LTE for Devices

FP7 Contract Number: 258512

WP4 – E2E M2M System

D4.5
Monitoring of Network Sensors and Actuators

Contractual Date of Delivery to the CEC:	
Actual Date of Delivery to the CEC:	
Responsible Beneficiary:	VGSL
Contributing Beneficiaries:	VGSL, TST
Estimated Person Months:	13
Security:	Public
Nature	Report
Version:	1.0

Document Information

Document ID:	Monitoring of Network Sensors and Actuators.doc
Version Date:	31 October 2012
Total Number of Pages:	
Abstract	The purpose of this deliverable is to define Packet Data Monitoring Architecture for Integrated Sensors and Actuators.
Keywords	EXALTED, M2M, Network Monitoring, Network Management, Wireless Sensor Networks, LTE-M, E2E M2M system, Capillary Network, Device Management, Resource Management.

Authors

Name	Organisation	Email
Mohammad Reza Akhavan	VGSL	reza.akhavan@vodafone.com
Tony Sammut (Editor)	VGSL	tony.sammut@vodafone.com
Javier Valiño	TST	jvalino@tst-sistemas.es
Juan Rico	TST	jrico@tst-sistemas.es
David Garcés	TST	dgarces@tst-sistemas.es
Daniel Getino	TST	dgetino@tst-sistemas.es

Approvals

	Name	Organisation	Date	Visa
Technical Manager	Pirabakaran Navaratnam	UNIS	31/10/2012	OK
Project Manager	Djelal Raouf	SWIR	31/10/2012	OK

Executive Summary

Network monitoring tools and techniques are required to support the deployment of monitoring network services and elements by providing the necessary information from the current status of the sensor nodes and network.

In this report, first the requirements derived from the use cases detailed within the EXALTED project have been collected and analysed from network monitoring point of view. The main requirements of network monitoring tools and techniques are reliability, adaptability, and scalability. In addition, a literature review has been conducted in order to determine the available approaches for network monitoring system architectures, comprising theoretical and actual implementations. The identified approaches are evaluated against the EXALTED requirements in order to identify their benefits and weaknesses. From the analysis, the hierarchical approach is recommended for network monitoring architectures.

The hierarchical framework enables the intermediate devices (Gateways, Cluster Heads) to combine their monitoring status with the results from the connected M2M devices and thereby capture the status of the network. Moreover, the hierarchical network monitoring framework tackles the scalability requirement by applying monitoring policies at different network levels as well as reduces the impact of traffic load on the LTE/LTE-M network.

In addition, it introduces the standard interfaces allowing interoperability with other device functionalities such as device management and self-diagnostics.

Finally a section regarding a real lightweight monitoring mechanism implementation for M2M devices is presented. This is a practical study which relies on the assumptions previously mentioned but putting the stress on testing the most suitable data transmission mechanisms already available. It provides the needed connection between the theoretical studies and what is implementable.

Table of Contents

1. Introduction

Wireless Sensor (and actuator) Networks (WSNs) comprise hardware/software-constrained sensor/actuator devices that are prone to failure due to network dynamicity and topology changes that may happen frequently. Nodes may be stolen from the network or destroyed, they may run out of battery power or be removed from the network due to failure in their hardware [2]. Additionally, new links may be removed or created due to node mobility. The dynamic nature of this type of network requires efficient mechanisms to monitor and manage the components [3].

The major challenges in designing network management protocols are scalability, energy-efficiency and reliability. Additionally, the data overhead generated as a result of monitoring needs to be minimised because the sensor devices have limited processing capability and memory.

This deliverable defines a packet data monitoring architecture for sensors and actuators within the EXALTED framework. This monitoring architecture is compliant with the EXALTED requirements and LTE-M system characteristics (i.e. reliable, scalable and configurable) and support the use cases and scenarios defined in [1].

Network management comprises *network monitoring* and *management control* [4] functions. Network monitoring tools and techniques are required by network elements and services for providing necessary information from the current status of the sensor nodes and network. The collected information at sensor node level includes: energy state (remaining energy, scavenging rate, and depletion rate), and memory state. Network-wide information includes, but is not limited to, connectivity, coverage, topology, and mobility. The network manager can initiate appropriate actions resulting from the network monitoring queries.

In the absence of a standard for the monitoring of wireless sensors, this report revisits existing management and monitoring approaches for wireless sensor networks. These approaches are extracted and classified in terms of their functional components and how they address the requirements of monitoring/management in a sensor network.

Monitoring information received from the status of the network integrated sensors and actuators enables the management server to make management control decisions and maintain/enhance the performance of the network. The main reasons for monitoring/management can be summarised as follows:

a) **Fault detection:** In many applications of network integrated sensors and actuators, failure of a few single sensors should not affect the overall operation of the network [4]. The acceptable level of failure in a network for a satisfactory level of operation is defined as *fault tolerance*, which is the ability to maintain sensor network operation without any interruption despite sensor device failure [5], [6], [7]. This level may vary greatly between different applications. It is the responsibility of network monitoring system to determine when, where and why a fault has occurred [8] and to provide as assessment the level of damage to management modules.

b) **Performance Optimization:** Network monitoring by discovering and providing the network inventory[1] and network health status can be used to maintain the satisfactory level of performance. For example network monitoring can monitor the remaining

[1] Network inventory represents number, type and status of devices on a network

battery and/or energy consumption rate at different sensor nodes in order to increase the life-time of the network and balance the energy level[2] in wireless sensor networks. Such a power management system can be achieved by changing their duty-cycle or changing the routes, route management, protocol update, and traffic management, sampling rate control, etc.

1.1 Types of Network Monitoring / Management

Network management systems have been categorised in different approaches with respect to monitoring and control [9], [4]:

- **Passive monitoring (time-driven):** Network monitoring continuously collects information at a constant rate from the network and analyses it at the sink (collection point). Based on this post processing approach, if any major change in the network status is detected the management control tasks will be triggered.

- **Reactive monitoring (event-driven):** Network monitoring is initiated at the sink if an event of interest occurs. In this approach the management solution is reliant on real-time processing of collected information.

- **Proactive monitoring (query-driven):** Network monitoring is performed actively based on real-time collected information from the network (user/server initiated queries). The management protocol reacts to any changes to predict and control any future events.

1.2 Network Monitoring Architectures

Network architectures can be categorised into three main areas depending on where the network the management operation resides [4], [9]:

- **Centralised:** In centralised management systems, such as BOSS [10], MOTE-VIEW [11], SNMS [12], and Sympathy [13], the central node (sink, gateway, or cluster head) acts as a network management functionality, collecting information from all nodes and controlling the entire network. A sink with no resource constraints can perform complex management decisions based on global information from the network. This reduces the processing load for resource-constrained sensor nodes. However, this approach has several disadvantages: Monitoring data must be transmitted from each sensor node in the network to the sink, which increases data overhead and thus limits its scalability. Since individual node information is important, aggregation solutions may not be applicable. In addition, in the case of network partitioning, the nodes that are unable to reach the central sink are left without any management support.

- **Distributed:** In this approach, several network managers are distributed in multiple nodes called cluster-heads (CH) which have higher capabilities and less resource constraints than sensor nodes (also called leaf nodes). Each manager controls a subnetwork and may communicate directly with other management stations in a cooperative way in order to perform management functions. This method improves the robustness of the solutions and decreases the communications cost, as nodes are controlled by local managers instead of a single centralised sink. However, it is a complex solution and coordination and associated communication between distributed network managers is a major challenge. In addition, processing load and

[2] In wireless sensor networks energy used in sensor nodes close to the sink (central collection point) is higher than other areas. This may cause early depletion of their battery and failure of these nodes, resulting disconnection of the sink from the rest of the network and consequently shorter lifetime of the network

memory cost are high at network managers, which either require specific hardware or result in limited network management functionalities. Distributed management systems include DSN RM [14], Node-energy level management [15], and AppSleep [16]. An Agent-based framework (such as Agilla [17], Sectoral Sweeper [18], Mobile Agent-Based Policy Management [19], and MANNA [20]) can be considered as an example of distributed monitoring system.

- **Hierarchical:** In hierarchical network management, tasks are distributed among network managers. This is similar to the distributed approach, however intermediate managers (e.g. CH) do not communicate with each other directly and each manager reports to a higher level manager, leading to the sink at the top level. Local management tasks can be performed at a lower level, reducing communication costs. Meanwhile a global view of the network can still be available by reporting lower-level managers to a higher level sink which can enable the sink to make network-wide management control decisions. AppSleep [16], STREAM [21], and SenOS [22], are examples of this class of network management.

1.3 EXALTED High Level System Architecture

The most important elements of the EXALTED system architecture are the following [23]:

A. **M2M devices**: Devices that can support one or more M2M applications. They can be categorised into:
   - **LTE-M Devices**: they implement both the proposed LTE-M interface and capillary interface technologies and can access the application domain, either by directly accessing the LTE-M network, or through an LTE-M Relay.
   - **Non-LTE-M Devices**: they do not have the LTE-M interface, but will form capillary network(s) using other network access technologies, such as Zigbee, and IEEE 802.11x. They can access the application domain through a M2M Gateway, or may run M2M applications locally.

B. **LTE-M Relay**: These elements are similar to 3GPP rel.10 LTE-A Relays. They are used in the LTE-M environment for coverage extension and throughput enhancement (through link diversity) They form a capillary network with LTE-M Devices, which use LTE-M Relays for communication with the rest of the network.

C. **Non-LTE-M Cluster Heads** (CH): They can be considered as more powerful M2M Devices with some additional capabilities, or as M2M Gateways with reduced functionality. Like regular M2M Devices, they are also part of capillary networks and the communication from a regular M2M Device may be directed through and managed by a CH. The functionalities of a CH may include data aggregation, device management, routing, etc. Unlike a M2M Gateway, a CH will not perform protocol translation.

D. **M2M Gateway**: Provides interconnection between the LTE-X (i.e. LTE/LTE-A/LTE-M) network and the capillary networks (comprising one or more M2M devices). It can provide various functionalities, such as protocol translation, routing, resource management, device management, data aggregation, etc. In some cases, the M2M Gateway may also act as an application server providing M2M services locally in the capillary network. It is expected that the M2M Gateway will normally connect to the LTE-X network with a direct radio link. In the case that an M2M Gateway is unable to establish direct connectivity (for example, due to deployment in a remote area without coverage, or due to localised infrastructure failure), then connectivity to the LTE-X network may be achieved by hopping via an LTE-M Relay and/or one or more other M2M Gateways. Some capillary network radio interfaces will support such M2M

Gateway to M2M Gateway links. Direct M2M Gateway to M2M Gateway connectivity for the purpose of E2E device to device connection without any LTE-X involvement (e.g. local breakout) is not the primary focus of EXALTED.

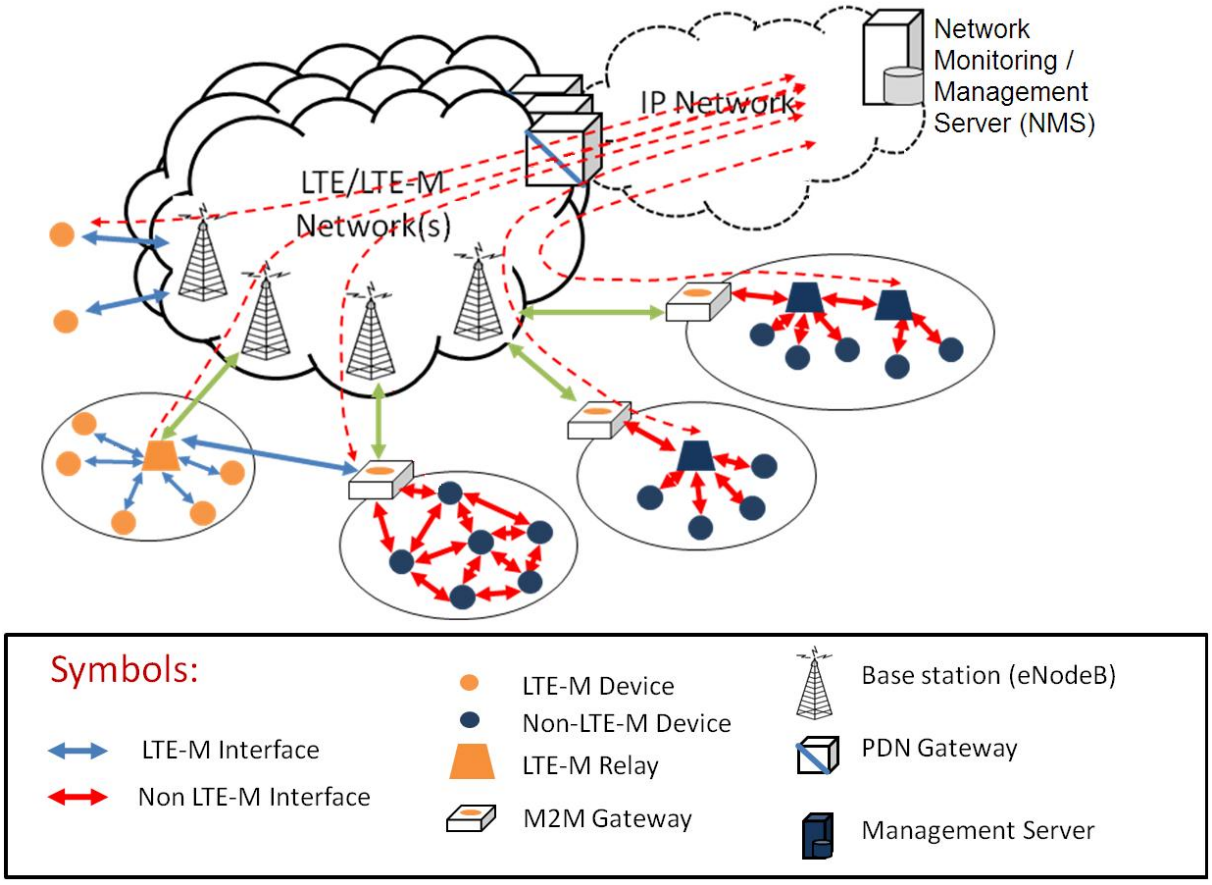

Figure 1.1 EXALTED high-level architecture

The core of the **M2M Network Domain** is the LTE/LTE-M system, involving all elements that can be found in LTE (e.g. eNodeB, EPC (Evolved Packet Core)), but with the necessary enhancements in order to support M2M communications. It is assumed that interconnection between different LTE/LTE-M system deployments will exist, such that M2M devices and application servers connected to different deployments will be able to discover each other and communicate as if they were connected to the same deployment.

On the top of the underlying protocols and technologies, M2M Application Servers communicate with M2M Devices and Gateways that support the same application. Note that the applications may run on any functional element (i.e. on the M2M Gateway, the LTE-M Relays, the M2M Devices, or the Cluster Heads). Apart from the Application Servers, the architecture assumes other relevant servers for management and control, such as Device Management Server, which uses specifically designed protocols over the same network for communication with Devices and Gateways. All of the abovementioned servers need to fulfil the security requirements (EXALTED WP5 Deliverable [24]).

2. Challenges and Requirements for Network Monitoring

This section explains the challenges for network management/monitoring in WSNs.

Due to the unique characteristics of WSNs, special considerations need to be addressed in designing management/monitoring schemes for this type of networks. Some of the challenges are [4]:

- **Data overload:** A large number of sensor nodes will increase the amount of collected information from the network. In addition, different types of data required by the network management function/solution may dramatically increase the information from each sensor. Since the communications are very costly in terms of energy in WSNs, management solutions should utilize minimum data (i.e. overhead) for better energy efficiency.

- **Unreliable communications:** Because the wireless links connecting the nodes in a WSN are unreliable by nature, high error rate, network congestion and interference are commonplace and may affect accurate monitoring of sensor devices.

- **Information visualisation:** The large amount of data continuously received from the network is a challenge when considering the processing required to analyse the network state.

Additional requirements for the monitoring framework for sensor network services are simplicity and extensibility. The framework should not be hardware-dependent and should support the changing or adding of new features, sensor network protocols, scenarios or other tools required by the users [25]. This framework should allow for high fidelity capturing of sensor network or sensor network application events [26]. In addition to these there are other general requirements that need to be taken into account such as: scalability, fault tolerance, and energy efficiency. Finally the incoming or outgoing network monitoring queries and commands (queries and session, etc.) must be authenticated and secured.

In the rest of this section the term “M2M device” is used to refer sensors and/or actuators in capillary networks.

2.1 EXALTED Requirements

This subsection restates the requirements for Monitoring of Network Integrated Sensors and Actuators derived from the use cases introduced in [27]. The different parameters involved are explained and a table of requirements is presented.

The study presented within this document is concerned with network health monitoring of sensors/actuators to ensure that there are on-line, bi-directional, demand-response communications between sensor-actuators and gateways. This task is different from Device Management in the way that it is not only focused on identifying non-functioning devices but monitoring the operation and maintenance of the network, which is the main objective.

Category	Description	Requirements		
Energy efficiency	LTE-M device, Sensor nodes and / or actuators may run on batteries. In most application scenarios changing the batteries are impossible or impractical therefore, energy efficiency is crucial in most LTE-M networks in order to increase the life time of the devices and network.	Network	Energy efficient routing protocol	NEEDED
			Energy efficient MAC protocol	NEEDED
		Device	Energy efficient OS (Software)	NEEDED
			Energy efficient processor (Hardware)	NEEDED
Reliability	Having acceptable level of reliability for most applications is very important. Notification of any failure to deliver the message is a key issue in Task 4.5. Protocols intended to be used for monitoring and auditing the devices must be reliable in order to maintain the satisfactory level of operation of the network.	Reliable delivery of a message		NEEDED
		Notification of any failure to deliver the message		NEEDED
		Notification of failure in demand-response communications between sensor and actuator		NEEDED
Deployment architecture	This group covers different topologies and how data is exchanged through them, within capillary network and also the relation with the gateway. There are many possibilities in topologies as well as in communication mode among nodes.	Device-to-device communication		NEEDED
		Infrastructure-to-device communication and vice versa		NEEDED
		Hybrid architecture, devices-to-cluster head		NEEDED
		Cluster head -to-infrastructure		NEEDED
		Multi-hop		NEEDED
Transmission mode	Depending on radio channel capabilities used by nodes for establishing communication, three different set of requirements can be set. Each mode supposes different characteristics in terms of available bandwidth, and power consumption.	Simplex		NEEDED
		Half-duplex		NEEDED
		Full-duplex		POSSIBLE
Communication type	The messages sent and communication links could be set in different manners depending on communication infrastructures and network implementation over nodes in capillary network. This group exposes the possibilities that the capillary network should offer to their members.	Peer-to-peer		NEEDED
		Unicast		NEEDED
		Multicast		NEEDED
		End to end communication		NEEDED
		Broadcast		NEEDED
Network partitioning	Depending on network size, in number of nodes and in terms of area covered, the network could be divided into smaller sectors, clusters, which improve the network behaviour in all aspects. This partitioning scheme allows devices to reduce power consumption and increase network efficiency.	Clustering		POSSIBLE
		Cluster-head selection		POSSIBLE
Transmission range	Parameters associated to this group show the needs in terms of proximity of the nodes between themselves and between them and the gateway. The range of communications could be from less than a few meters to more than hundreds, depending on the scenario, output power and physical characteristics of the implemented antennas.	Short Range (<1m)		NEEDED
		Medium Range (<100m)		NEEDED
		Large Range (>100m)		NEEDED
Node density	The number of directly connected devices to the gateway or cluster head. Depending on the application scenario, the number of deployed devices in a geographical area may vary.	High		NEEDED
		Average		NEEDED
		Low		NEEDED
Device capabilities	Under this categorisation, the hardware and standard-dependant parameters will be studied, such as the cost, power consumption, data rate, etc.	Power	Battery-powered	NEEDED
			Main-powered	POSSIBLE
		Data rate	Low data rate	NEEDED
			High data rate	IRRELEVANT
		Cost	Low	NEEDED
			High	POSSIBLE
Gateway	It is one of the most critical parameters on the network as well as one of the most scenario-dependant devices.	Address translation (between LTE-M and capillary network)		NEEDED

capabilities		Power consumption (Batteries or plugged)	Battery-powered	POSSIBLE
			Main-powered	NEEDED
		Maximum supported devices (Scalability)		NEEDED
Mobility	Depending on application scenario, different mobility patterns could be identified: device mobility, group mobility, and / or gateway mobility. Mobility may have an effect on other requirements such as communication range, network partitioning, and reliability. Due to mobility existent communication links may break or new links may create which lead to topology changes	Single user mobility		NEEDED
		Group mobility		NEEDED
		Nodes are static and sinks are mobile		NEEDED
		Nodes are mobile and sinks are static		NEEDED
		Both nodes and sinks are mobile		NEEDED
		Adaptive mechanisms/parameters for Mobility Management		NEEDED
Addressing scheme	Different addressing schemes are necessary in capillary network and LTE-M device.	IEEE (ZigBee, DASH7…)		POSSIBLE
		IP	IPv4	POSSIBLE
			IPv6	NEEDED
		Basic MAC address		POSSIBLE
Delay / Real-time interactions	Different use cases demand a different level of delay. For instance in traffic optimisation, delay is very important while protocols used for environmental monitoring can be delay-tolerant	Real-time/emergency/safety applications		NEEDED
		Delay-tolerant		POSSIBLE
Security	Security is required to prevent eavesdropping, hackers' attacks including IP-based attacks.	MAC level		POSSIBLE
		Network level		NEEDED
		Application level		NEEDED
Location information	Exact or relative position information of the devices is necessary for some application scenarios. Position information can be provided by GPS system installed in each device. In wireless sensor networks with low-cost, low-energy sensor nodes may obtain their relative location by means of an anchor node	GPS		NEEDED
		Anchor node		POSSIBLE
Reporting pattern	Gateway in capillary sensor network may receive the sensed data from the nodes in fixed and predefined time interval. In some applications periodic reporting may not be considered as an energy-efficient approach, sensors or actuators may operate when they detect an event or receive a query from the gateway.	Event-driven reporting		NEEDED
		Periodic reporting		NEEDED
		Query-based		NEEDED
		Combined solution		NEEDED
Continuous connectivity	Many real-time or critical applications demand continuous connectivity.	For real-time applications		NEEDED
		For critical reporting		NEEDED
		For others		POSSIBLE
Data rate	In EXALTED project assumed that network devices generate small to average data-rates.	High data rate		OUT OF SCOPE
		Low data rate		NEEDED
		Average data rate		POSSSIBLE
Scalability	Depending on the applications, large number of devices could deploy in the capillary network. The system and communication protocols need to be capable of handling increased number of nodes.	Addressability		NEEDED
		Routing protocol		NEEDED
		Cluster head / Gateway		NEEDED
Traffic pattern	Three main traffic flows for most capillary networks are follows:1) Query requests from the sink (s) towards the sensor nodes 2) Reply (sensed information) from the sensor nodes towards one or multiple sinks 3) Control commands from the sink(s) towards the actuators.	Many-to-one		NEEDED
		One-to-many		NEEDED
		Highly directed		NEEDED
Server initiated communications	Remote-monitoring applications for the many scenarios require server initiated communications, as they may be based on a data polling scheme, from a central server to the monitoring devices. However, some other applications such as Car-to-Car (C2C) communications may not need this kind of operation.			NEEDED

The communications protocols for monitoring of network integrated devices should satisfy the following requirements:

- **Communication range:** Depending on different scenarios and applications, the transmission range between devices and between device and gateway within a capillary network may vary from a few meters to more than hundreds of meters. Some parameters that affect the transmission range are: wireless technology, physical characteristics of the antennas (for example power, length, etc.).
- **Node degree/density:** Depending on the application scenario, the number of deployed devices in a geographical area may vary. The number of directly connected devices to a gateway or cluster head could also vary.
- **Location information:** Exact or relative position information of the devices is necessary for some application scenarios. For example, in environmental monitoring the sensed data needs to be mapped by location of the sensor nodes and in traffic optimisation it is important to have each vehicle's position, speed and current trajectory. Position information can be provided by a GPS system installed in each device. However in a capillary networks, low-cost, low-power M2M devices may obtain their relative location by means of an anchor node (i.e. a node with known location).
- **Mobility:** Depending on the application scenario, different mobility patterns could be identified: device mobility, group mobility, and/or gateway mobility. Mobility may have an effect on other requirements such as communication range, network partitioning, and reliability. Due to mobility existent communication links may break or new links may create which may lead to topology changes.
- **Reporting pattern (Event-driven, Periodic or Query-based):** A gateway in a capillary network may receive the sensed data from the nodes in fixed and predefined time interval. In some applications periodic reporting may not be considered as an energy-efficient approach, sensors or actuators may operate when they detect an event or receive a query from the gateway.
- **Energy efficiency/Battery life:** Sensor nodes and/or actuators may run on batteries. In many applications, if not most, changing the batteries is impossible or impractical. Energy efficiency is therefore crucial in most capillary sensor networks.
- **Security:** Security is required to prevent eavesdropping and hackers' attacks including IP-based attacks.
- **Continuous connectivity:** Many real-time or critical applications demand continuous connectivity.
- **Topology:** (Single-hop / Multi-hop) nodes may be out of communication range of the gateway. Multi-hop communications in this case can extend the coverage of the network and increase energy efficiency because there is no need to increase the transmission power.
- **Scalability:** Depending on the application, large number of devices could be deployed in a capillary network. The system and communication protocols need to be capable of handling increased numbers of nodes.
- **Reliability:** Having an acceptable level of reliability for applications, such as security and e-Health, is very important. Protocols intended to be used in EXALTED for monitoring and auditing the devices must be reliable in order to maintain a satisfactory level of network operation.
- **Hybrid topologies:** In some application scenarios, large number of devices can be partitioned in small groups (clusters) in order to increase communication and management efficiency.
- **Data rates:** In the EXALTED project it is assumed that network devices generate small to average data-rates.

2.2 State of the Art

2.2.1 Bridge of Sensors (BOSS)

Song et al. [10] proposes BOSS (Bridge Of the SensorS), a service discovery management architecture for remote management of wireless sensor networks implementing the Universal Plug and Play (UPnP) protocol. UPnP requires high computation power and large memory; therefore resource-constrained sensor nodes are unlikely to process the UPnP protocol [9]. Song et al. were limited to implementing BOSS using resource-rich devices. The network management services supported by BOSS are basic network information, localisation, synchronisation, and power management. The advantage of BOSS is that it can adapt to topology changes and support proactive network management; however the main disadvantage is that it needs an end-user to observe network states and take management actions accordingly [9]. In short, although BOSS is energy efficient, robust, adaptable, and memory efficient, it is not scalable.

2.2.2 Management Architecture for Wireless Sensor Networks (MANNA)

Ruiz et al [20] has proposed MANNA, a centralised management architecture that dynamically collects information from sensor nodes and maps this information into WSN models/maps. The relationship among these WSN maps is defined in a Management Information Base (MIB). The followings are some examples of WSN maps:

- **Sensing coverage area map:** This map describes sensing coverage of the sensors
- **Residual energy map:** Provides a topological view of battery level of sensor nodes in the network
- **Communications coverage map:** Provides network connectivity by using communication range of sensor nodes and their neighbours

A distributed architecture requires higher processing for nodes with agent duties.This increases the processing and memory burden at the nodes [4].

2.2.3 Simple Network Management Protocol (SNMP)

SNMP is a component of the Internet Protocol suite as defined by the Internet Engineering Task Force (IETF). It has been widely used in IP-based networks as a network management protocol for monitoring devices connected to a network. SNMP comprises an application layer protocol, a database schema, and a set of data objects [28].

SNMPv3 defines seven core protocol data units (PDUs); five core PDUs plus *GetBulkRequest* and *InformRequest.* Simplicity and wide deployment of SNMP are the main benefits. In addition, SNMP only manages network elements rather than supporting network-level management.

2.2.4 Sensor Network Management System (SNMS)

SNMS is an interactive system for monitoring the health of sensor networks proposed by Tolle et al. [12]. SNMS can work in two different configurations: query-based and event-driven. It forms a gathering tree to disseminate the queries and collect the health data. The advantage of the query-based approach is that it is independent from the application and can provide management functions. SNMS has the following benefits and drawbacks [9]: it introduces overhead only for human (ad-hoc) queries and thereby reduces network traffic and the need for memory. It saves energy by bundling the results of multiple queries into a

single message instead of returning results individually. However, it is limited to passive monitoring only, requiring manual management, submission of queries and analysis of the gathered data. The centralised-processing approach has the positive impact of extending the lifetime of the network. In short, SNMS is energy and memory-efficient, but is not adaptive and scalable.

2.3 High Level Description of Network Monitoring

As stated in previous sections, network monitoring is part of network management. The network manager can initiate appropriate actions resulting from the network monitoring queries. The solution proposed for network monitoring tackles the scalability requirements by introducing a hierarchical architecture in which there is a monitoring module in each LTE-M device and/or M2M device. This module monitors the latest status of predefined parameters in the devices. (e.g. battery level, wake up duty cycle, sample rate, etc.). At the higher level of this hierarchy, there are monitoring/management modules in cluster heads (CHs), gateways (GWs), or eNodeBs (eNBs). Each CH, GW or eNB is responsible for applying network monitoring/management policies for their leaf nodes. These policies are defined though the network monitoring/management server (NMS) [Figure 1.1]. For example the NMS can define “energy policy” for a particular capillary network requesting the statistics for a specific resource (e.g. leaf nodes with less than 30% energy level in percentage). NMS may need this information to increase the performance of the network, for load balancing or changing the duty cycles of the network to increase the life time of the sensor network. NMS could define the policy to the intermediate managers (CHs, GWs, or eNBs) to initiate appropriate actions resulting based on the predefined policies. For instance NMS can ask the intermediate managers to change the duty cycle or reporting period of the M2M devices if their energy level drops below a certain threshold. This rule-based system with local management at different level of the hierarchy will dramatically reduce the traffic load over the radio link to the core network, saving energy and increasing the scalability of the operation because there is no need to relay the status of the individual nodes to the NMS server, intermediate devices can operate on their own and apply the predefined policies from the NMS server as long as they receive a new policy/rule from NMS and M2M devices only respond to the intermediate devices with requested monitoring information.

2.3.1 Collaboration with Device Management and Self-diagnostic Modules

Collaboration between different modules in EXALTED can optimise the shared resources and reduce data exchange, energy consumption and memory and processing usage. Among different modules in EXALTED, device management (EXALTED WP4 Deliverable D4.3 [29]) and self-diagnostics (EXALTED WP6 Deliverable D6.3 [30]) are the most relevant modules to network monitoring. A Service Oriented design [30] is adopted with well defined interfaces and subscription mechanism for ensuring and optimizing remote device management access to various types of devices. This generic approach to design makes interconnection easier.

Each M2M device in the EXALTED solution is expected to have a client for device management [1]. The local device management client can provide some services to the network monitoring and self-diagnostic modules as follows [30]:

- NMM and SDM components provisioning (rule set download at device installation…)
- Identification (provisioning servers)
- Authorization (rule activation and execution, subscriptions…)
- Files and data transfers

On the other hand self-diagnostic aims to [30]:

- Locally proceed (as possible) to a diagnostic of its own status (hardware and software) to avoid maintenance traffic
- Ability to execute autonomous diagnostic (called synchronous) or/and remote diagnostic (called asynchronous) and to return the interpreted result to requester
- Ability to notify (if expected) the self diagnostic results – regarding the priority of the service - to its subscribers

- Ability to proceed to corrective actions – introduced by device management or network monitoring modules - or modify current configuration in order to fix behaviour under prior authorization conditions
- Storing as possible the diagnostic history traces

The inclusion of a Self-diagnostic module (SDM) in the M2M device has been considered optional [30]. However the output of the SDM module will be more than sufficient upon which to support the NMM.

In the architecture presented below, the M2M Device must have either NMM or SDM modules and the NMM module in Gateway/Cluster Head has interaction with the relevant DM/SDM modules. The device management interface client provides security services to the NMM and SDM modules.

Based on the above described functions for the SDM, it is expected that SDM in the M2M device can provide the information required by higher level network monitoring module as illustrated in Figure 2.1. Therefore the existence of one of these modules would be enough to support the NMM functionality. This can be done by adopting the SDM architecture [30] in designing NMM for M2M devices.

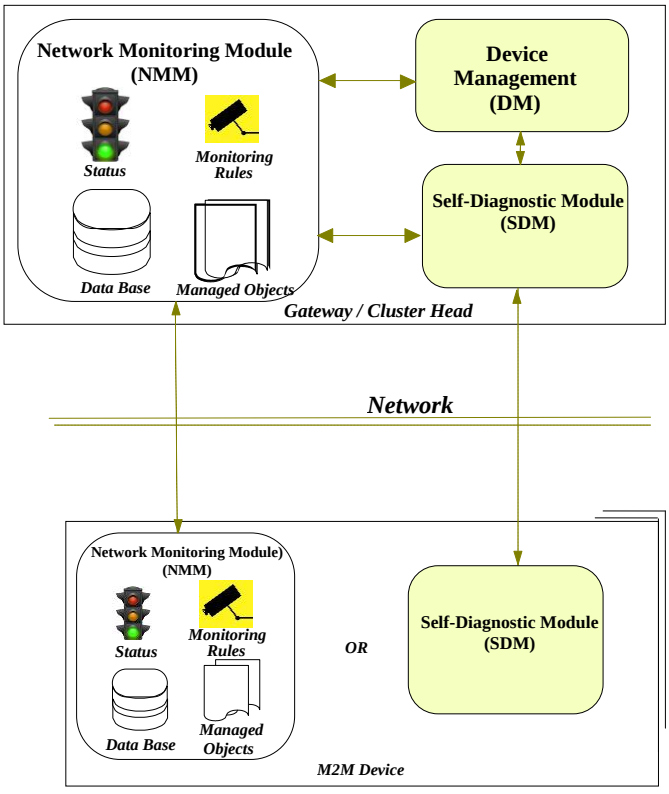

Figure 2.1: Network monitoring modules and interfaces

In this section a software framework based on Authority Oriented Client-Server (Master-Slave) [31] is proposed in which master/client can be a monitoring module in upper level (GW, CH or eNB) and the slave/server can be another monitoring module or self-diagnostic module in the M2M device. In the hierarchical architecture (explained in section 1.1), each NMM is slave for the upper level associated module and is master for the lower level associated modules. In this recursive topology each NMM module has its own authority level

and is responsible to report/respond to the higher level and directs queries towards lower level modules.

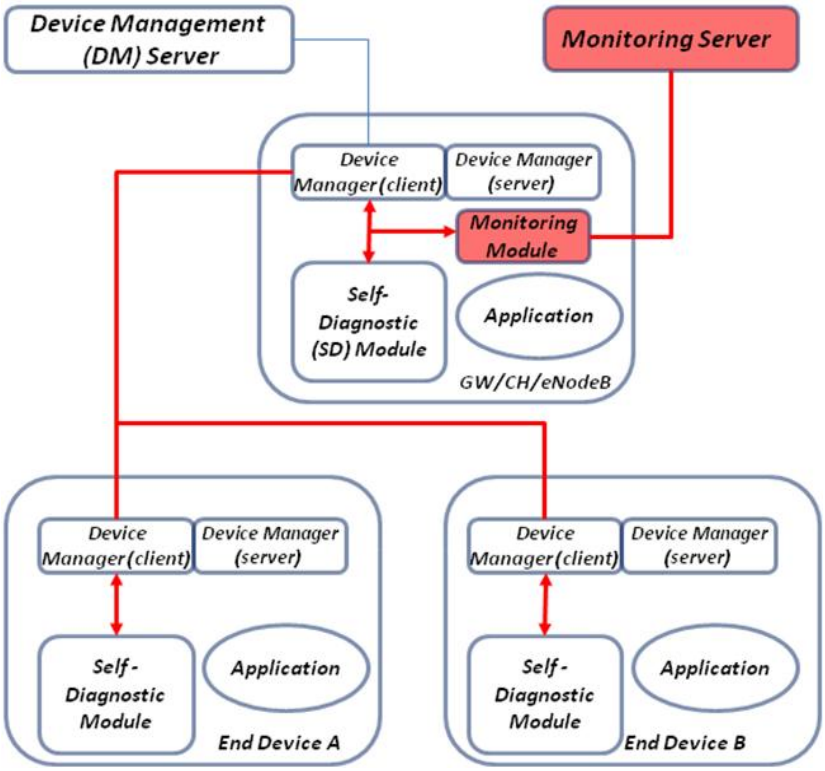

Figure 2.2: Network monitoring - a hierachical architecture

3. Network Monitoring Architecture for EXALTED

3.1 NMM Internal Architecture

A common architecture framework for network monitoring and self-diagnostics has been defined in Network Monitoring, Device Management [29] and, Self-diagnostics [30] within EXALTED. A similar architecture introduced for SDM is adopted here with some modifications and amendments to be applicable to network monitoring. As previously outlined (section 2.3.1), the main reason for this adoption is the potential of using SDM in the M2M device instead of the NMM. However, since the SDM module is optional in EXALTED [30], a similar architecture for NMM in M2M device to operate comparable to SDM module will be used.

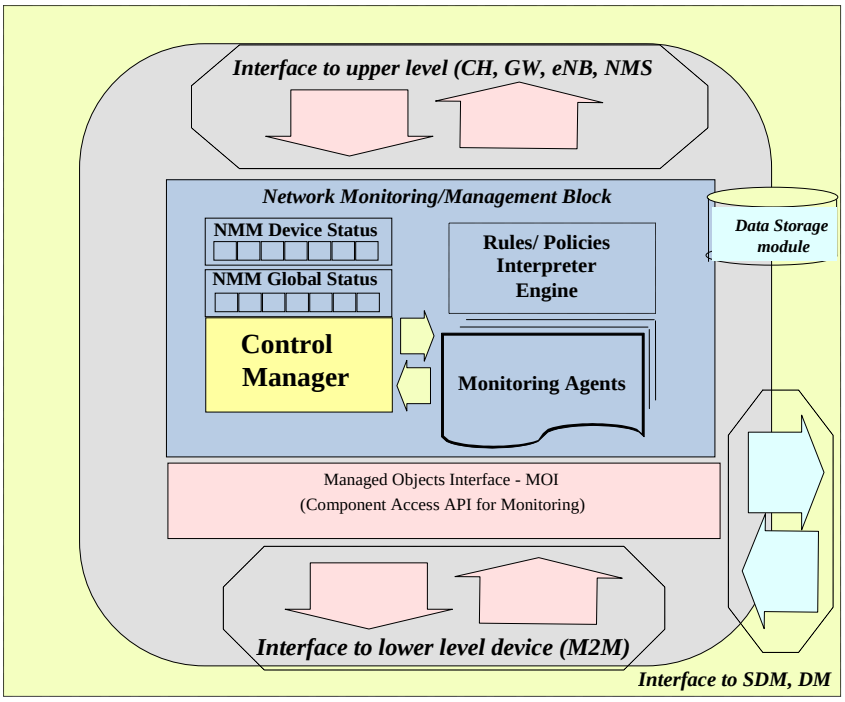

Figure 6.1: Internal structure of NMM for intermediate devices (CH, GW, eNB)

The most important building blocks of the NMM illustrated in Figure 6.1 are the following:

- Interface to upper level devices: an API that exposes NMM services to applications that are local to the device (device management client, etc.) or even external (remote device management, network monitoring, other NMMs entities)
- Network monitoring block: this component holds the monitoring logic
- The Managed Objects Interface (MOI): an interface to access the managed objects in the device

The network monitoring block consists of several components:

- Monitoring agents: these are instances of the monitoring policy applications. Initiating a specific monitoring is equivalent to invoking a specific monitoring agent.
- Policy/Rule interpreter engine: this module is responsible for the execution of the rules
- Decision manager: this module is responsible for the execution of the actions described in the rules that result from the monitoring

- NMM global status: this module provides a quick access to the current status of the device or the device it handles in collaborative or recursive monitoring modes

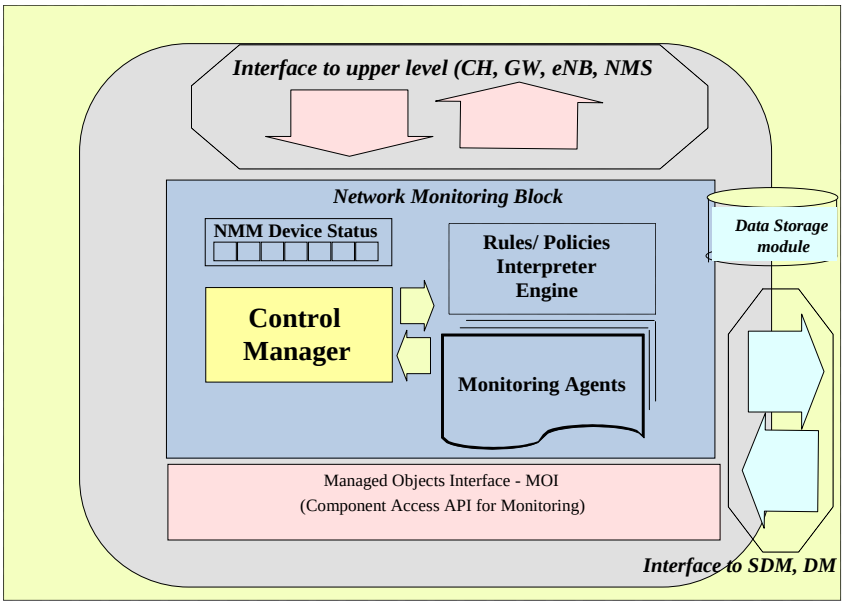

Figure 6.2: Internal structure of NMM for M2M device

The managed objects are connected to the entities they represent through an abstraction layer. This abstraction layer can be a hardware one or even a software one if the managed object represents software applications.

3.2 NMM Device Status / NMM Global Status

The NMM Device Status (NDS) stores and represents the current state of the device (M2M, CH, GW, etc). This information has been collected from managed objects in the device and is available to other local or remote clients subject to authorised access. The internal data structure and the content of the NDS are configurable. Depending on the device capability and complexity, the NDS is extensible and can integrate more or less information.

Similar to NMM Status, the NMM Global Status (NGS) contains the aggregation of the status of the lower level devices (NDSs) managed by this device. Normally NGS is placed in the CH, GW, or eNB and these intermediate devices update the NGS on a regular (periodic) or intermittent (on-demand) basis. NGS is an optional module in NMM and only exists in intermediate devices with more energy/hardware resources.

The NDS and NGS lists the identities of the Managed Objects (local or remote) used in monitoring functions. The information included in the NDS and NGS is available to any monitoring client, remote or local, that needs to identify the device's internals components or its relationship with other devices when collaborative or recursive monitoring mode are activated. Details of the NDS and NGS are available in Table 3-1.

Table 3-1: Details of NDS and NGS

NDS and NGS	Item description
Device information	
Device type	M2M device (Sensor, Actuator Cluster Head), M2M Gateway
PowerMech	Connected to main, Battery powered
MobilityLevel	Fixed, low mobility, high mobility
DutyCycle	Duty cycle of the node (radio is ON)
GeneratedTraffic	Amount of traffic generated per second
LastMonTime	Last reported time the monitoring status
ResEnergy	Residual energy level of the device
DepRate	Energy depletion rate of each node (average)
HarvRate	Energy harvesting rate of a node (i.e. for solar-powered node)
RepPeriod	Reporting period
Monitoring related information	
NMM version	The detailed information of current version
ActivationMode	Stopped, activated (NMM runs), deactivated (NMM suspended)
DefaultMonPeriod	Period of automatic monitoring execution

3.3 Monitoring Agents

The Monitoring Agents (MA) consists of different entities that are necessary to produce monitoring and take necessary actions advised from higher level nodes. MA deals with the results of the actions.

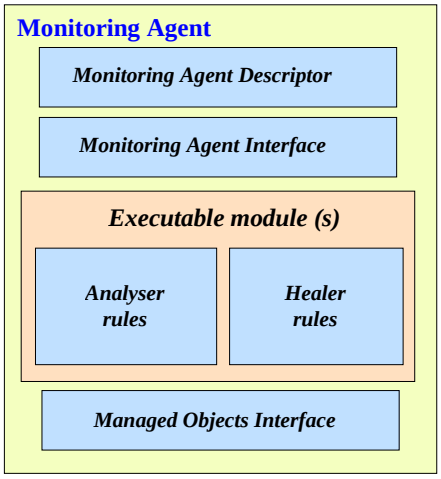

Figure 6.3: Monitoring Agent

The monitoring agent has similar components to the self-diagnostic agent [30]. Each NMM module can have one or more MAs and all MAs are interoperating with each other. They can also interact with managed objects. Figure 6.3 illustrates the main components of the MA.

3.4 Managed Objects

The managed objects are organised as defined in the OMA device management standard in order to share resources with entities such as local device management and local self-

diagnostic clients. The resources associated to the Managed Objects also hold their own access methods as defined in OMA-DM [32]. OMA-DM defines standard access methods: Get, Replace, Exec, Add, Delete, Copy. Table 3-2 shows the NMM specific behaviour related to the OMA DM access methods:

Table 3-2: OMA-DM access methods applicable to NMM [30]

OMA-DM methods	Executable node	Data node	Attributes
Get	Returns the node sub-net structure	Returns the first leaf level under the node	Returns the value of the attribute
Replace	Inactive	Inactive	Updates the attribute value
Exec	Runs the resource using its currents parameters defined in the node's attributes	Inactive	Inactive
Add	Accepted only from Local Device Management subscriber otherwise inactive	Accepted only from Local Device Management subscriber otherwise inactive	Accepted only from local device management subscriber otherwise inactive
Delete	Accepted only from Local Device Management subscriber otherwise inactive	Accepted only from Local Device Management subscriber otherwise inactive	Accepted only from Local Device Management subscriber otherwise inactive
Copy	Accepted only from Local Device Management subscriber otherwise inactive	Accepted only from Local Device Management subscriber otherwise inactive	Accepted only from Local Device Management subscriber otherwise inactive

3.5 Other Modules and Interfaces

Control Manager
The Control Manager is a functional block that administrates and ensures the execution of commands directed from upper level in the device. It can have different sub modules (e.g. Log Manager, Session Manager, Configuration Manager, etc.) It also ensures the interoperability with external nodes.

Policy/Rule Interpreter Engine
The Policy/Rules Interpreter Engine is used to hold the logic of the monitoring policies and rules and the monitoring rules. It processes the rules for managed objects.

NMM external Interfaces
NMM has external interfaces to allow authorised clients get access to the NMM services such as specific monitoring requesting, managed objects and rules management (creation, update, deletion, activation, deactivation). It also can be used to connect the remote NMM in remote device.

3.6 Monitoring and Management Parameters

Queries from Network:
The solution proposed in this study reflects a hierarchical monitoring architecture, where management/monitoring modules are deployed in the network in a hierarchical manner. Central network management/monitoring is also supported by assigning the management/monitoring tasks to the remote server. Similar to MANNA [20], different maps must be created. These maps provide a network-wide view of certain parameters in a network and its status (e.g. communication coverage map, residual energy map, etc.). On the other hand creating these maps based on separate queries increases data overhead. Therefore, information common to other modules (Device Management and Self-diagnostic) must be used. Below a list of the queries related to Device Management and Self-Diagnostics is presented which are considered to be necessary to support the DM, SDM and NMM modules in EXALTED.

List of Queries Common with Device Management:

- Type of M2M device (sensor, actuator, CH) or M2M GW)
- Device ID (or address)
- Status of device (DM-active, DM-idle, sleep, DM-sleep pattern)
- Number of children
- Duty cycle of a node (node's radio interface in ON)
- Amount of traffic generated by each node/CH/GW

List of Queries Common with Self-Diagnostics:

- Monitoring in other GW/CH - CH or GW connected to GW
- Energy mechanism (battery-powered, solar-powered, etc.)
- Residual energy level of the end device
- Energy depletion rate of each node (average)
- Energy scavenging rate of a node (i.e. for solar-powered node)
- Regular reporting device? (node can only send/receive data only during predefined time intervals)
- Event-driven device? (sensors send the report as the events happens – still may sense the environment periodically)
- Can device delay its data transfer?
- Device generating urgent/priority message/alarm? Device must be able to generate urgent/priority message beyond any restrictions (time, location, etc.)
- Is it a device requires secure connection?
- Received signal strength
- Average / maximum number of re-transmits for a packet

List of Queries Common with Device Management and Self-Diagnostics:

- Disk/storage capacity status
- Location information (of a device)
- Mobility (static, low / high mobility, move infrequently, or move only within a certain area)
- Wireless interface (ZigBee, Bluetooth, LTE-M interface, etc) DM (individual)-SD(for statistic)
- DL/UL Channel Quality Indicator:
- Sampling period (how often sensor node sense the environment)

4. Enabling Light-weight Monitoring Mechanisms on M2M Devices

There are two conflicting requirements derived from EXALTED's goals in the area of monitoring. On the one hand, device management and monitoring of M2M devices requires the execution of complex algorithms and transmission of large packets, so as to exploit available capabilities. This situation is due to the fact that existing monitoring systems are conceived for conventional and non power constrained equipment.

On the other hand, EXALTED deals with lower power and resource limited devices than the ones present on conventional deployments. Thus, there is the need to evaluate a trade-off between these two aspects.

There are several barriers to overcome in order to be able to properly monitor such devices:

- It would not be possible to transmit common monitoring files, they are too big and too complex to be embedded on the device's memory.
- It is not possible to assume that every node is IP-enabled, as stated when scenarios and use cases were defined [27].

This way, the proposed approach is to select simpler file transmission mechanisms, and Gateway delegated IP communications in case the node is not IP-enabled.

In addition, a socket-based address translation mechanism will be embedded and implemented at some point inside the capillary network (at least on M2M Gateways), enabling the monitoring feature. A real implementation in existing equipment is possible as well due to the fact of partner's hardware availability on EXALTED project [33].

4.1 Overview of the M2M Scenario

As introduced in the introduction of chapter 4, the baseline scenario intended to be addressed with this study is composed by the following elements:

- Non-IP, low power, low cost and low processor capability nodes, able to communicate with each other and the Gateway via a capillary (non LTE-M) interface, establishing dynamic and self-healing networks.
- IP capable Gateways, in charge of collecting data from the sensors within the coverage area and posting it in the M2M server via the LTE-M interface. They can also act as standalone devices posting their own data directly.
- A third party M2M Server, in charge of collecting, analyzing and displaying the received data.
- An M2M oriented middleware is used for monitoring data from sensors.

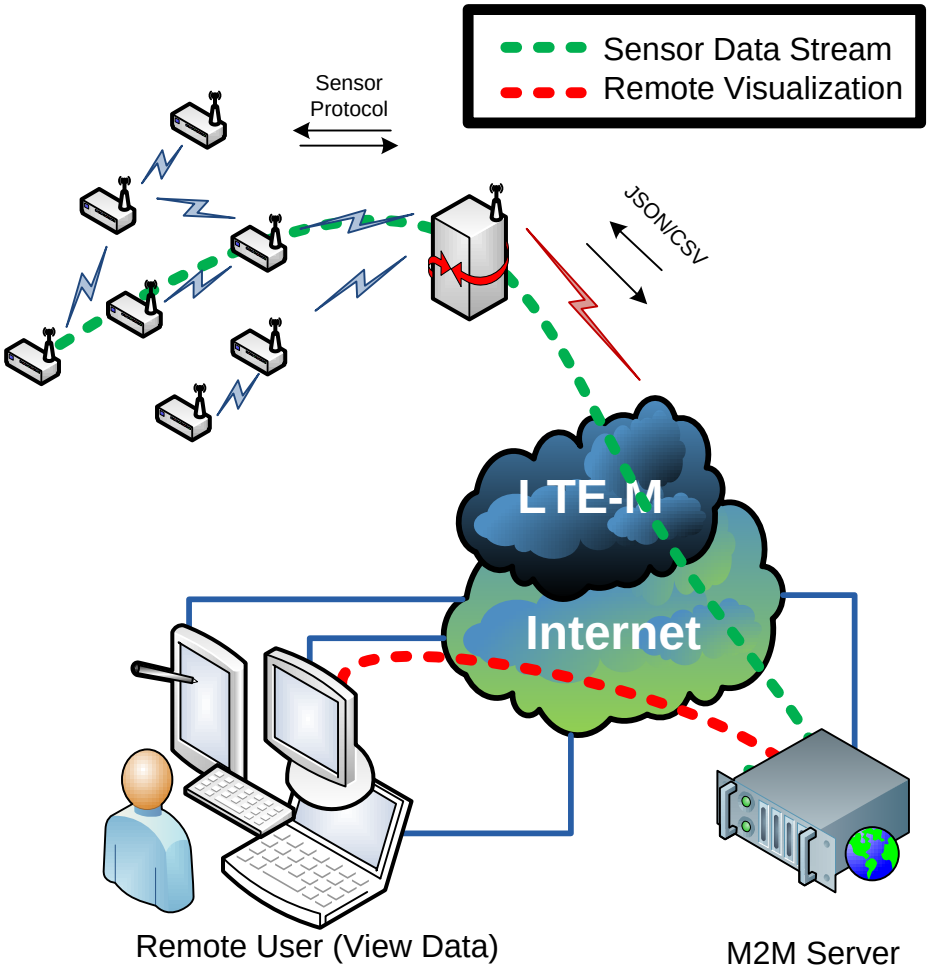

Figure 4.1: Basic architecture of the scenario

Given these initial considerations, the functionalities needed to be programmed on each of the devices are the following:

- Non-LTE-M M2M devices are responsible for transmitting their data towards the gateway. The data format chosen to send this information is designed to minimize the impact in terms of memory needed and radio usage.
- M2M Gateways (or IP enabled M2M devices) implement an HTTP client in charge of posting the information from the network every time a packet is received from the nodes. It is also possible to integrate aggregation techniques, implementing them combined with the solution proposed in this section. More details about these can be seen in [34].
- The M2M server runs a web server accessible for Gateways to send data from monitored devices. In addition, the server also offers a UFI (User Friendly Interface) enabling the visualization of historical and real time data, available for remote users to connect.

As stated on the Gateway functionalities, HTTP can be a protocol chosen for the high level interchange of information between the nodes and the M2M server. Several reasons are behind this selection:

- HTTP is a well-known and widely used protocol for posting remote information.
- Most of commercially available middleware solutions for M2M monitoring are based on this protocol.
- The implementation on the client side (end nodes) is really light and easy.
- HTTP support is natively supported on most programming languages.

It is true there are some disadvantages as it is not a bandwidth efficient way of conveying data. However from the perspective of M2M devices, it is a suitable approach in order to embed easy IP transmission capabilities with low resources and acceptable overhead.

By using HTTP, several ways to send data are supported. Depending on the application (the amount of overhead permitted, the flexibility at the server side, etc.), two methods can be used:

- "PUT" puts a page at a specific URL. The disadvantage is that, at server side, the previous info is deleted and replaced with the new information (or created if no previous info is available).
- By using "POST" method, the info can be processed at the server side. The server on the provided URL can do whatever it wants. For instance, the application programmed to test this study includes the ability to perform the following operations:
    - It can store it somewhere private. (HTTP 204 NO CONTENT).
    - It can store it in the page at the URL that was POSTed to (HTTP 205 RESET CONTENT).
    - It can store it in a new page, (returning the URL of that page) (HTTP 201 CREATED).
    - It can use it as input for several different existing and new pages.
    - It can insert, update, or delete records in a database (or all of the above).

Finally, the last variable to be fixed for the scenario is the data format. Traditionally, XML has been used for monitoring pieces of hardware due to its powerful way to include structures in a format which is both human and machine readable.

The disadvantage is that it requires multiple headers and tags to identify elements within the structures, which results in additional overhead when transmitting data.

When M2M low power devices are considered data overhead is a problem, as it means larger packets and transmission periods. As a result, other protocols have been defined to overcome these difficulties and facilitate monitoring for constrained devices:

- **JSON**. JSON stands for (Javascript Object Notation). It was introduced in 2001, but it did not become popular until Yahoo and Google popularized its usage around 2005. As its name implies, it was also created to more easily parse data into native Javascript objects, making it very useful for web applications. It maintains the hierarchical philosophy of XML but reduces the number of tags required. While XML needs an opening and closing tag for each element, JSON just uses one:

XML	JSON
<node> <id> Node1 </id> <value> 26 </value> </node>	{"id":"Node1","value":"26"}

- **CSV.** CSV stands for "comma separated values". As the name implies, this data format is basically a list of elements separated by commas. This is the most compact but format but it lacks versatility. The information transmitted is reduced to the minimum, but it requires parsing capabilities at the server side and synchronized updates (client/server) in case the format of the data is changed.

XML	CSV
<node> <id> Node1 </id> <value> 26 </value> </node>	Node1;26,...

4.2 Middleware Selection

As introduced in section 4.1, there are several options for middlewares which are commercially available for monitoring purposes.

To test the assumptions presented in section 4.1, the option that addresses the following requirements is selected:

- Open source.
- Widely used and tested.
- Compatibility with the most common nodes' programming languages.
- Light-weight client.

Given these constraints, there is a clear candidate for the implementation: COSM [35] (previously known as PACHUBE).

COSM is a data brokerage platform, which made a list of the top 5 trends in 2009 for the Internet of Things, managing millions of data-points per day from thousands of individuals, organisations and companies around the world. It enables storage, sharing and discovery of real-time sensors, energy and environment data from objects, and devices and buildings around the world.

The main features of COSM are as follows:

- Ability to send data in real-time
- Powerful API
- Previously developed applications
- Access to the developer community's data

The selection has been made due to some advantages that it presents related to the previously stated problem:

- Possibility of using and building applications which update and respond in real-time.

- CSV and JSON format available to send and receive data.
- Huge savings in terms of data transmitted compared to XML.

There are some other alternatives available, both free (e.g. ThingSpeak platform [36]) and proprietary (e.g. DEXMA [37], but they are not as versatile or as widely used as COSM.

4.3 Implementation on Real Equipment

Once selected all elements the monitoring scenario remain to be tested.,The required programming and modifications needed to be performed in each device are presented.

The characteristics of the elements described in previous sections are as follows:

- **END NODES**: 32 bit microprocessor with 42 KB RAM and 256 KB ROM nodes. Capillary radio interface. Temperature, Humidity and Accelerometer sensors.
- **GATEWAYS**: 32 bit microprocessor with 64 KB RAM and 1024 KB ROM nodes. Capillary and LTE-M radio interfaces. HTML client enabled.
- **M2M SERVER**: COSM
- **PROTOCOL**: HTTP PUT/POST
- **DATA FORMAT**: JSON/CSV

In an attempt to provide a broad view of all possibilities regarding light-weight monitoring, two clients have been developed for testing JSON and CSV formats, as well as using both POST and PUT methods.

For both examples, the behaviour at the gateway level is described. End nodes have been designed to send their data periodically. The data composes the readings of each of their embedded sensors (temperature, humidity and 3-axis accelerometer sensors) as well as useful data related to battery levels and network layout. Every time a packet is received by the Gateway, it must place a request to send this data to the M2M server. The implementation of the request and the data format are described in sections 4.3.1 and 4.3.2.

4.3.1 JSON

This first example, as explained in section 4.1, enables the definition of hierarchical packets and complex ways to process the information. This way, the HTTP method selected is POST. The needed header is then created using the following parameters.

```
/* POST HTTP client: header */
POST /v2/feeds/FEED_ID
Host: www.api.cosm.com/v2
X-ApiKey: API_KEY
Content-Length: LENGTH_JSON_DATA
Content-Type: application/x-www-form-urlencoded
Accepted: text/plain
```

Where FEED_ID is the identifier provided by COSM for the creation of the new data feed, and API_KEY the secure sharing key created for the transaction.

In order to finalize the request, the data must follow the header. In this particular case, the format chosen for sending the data is the following one.

```
“data”: JSONObject {
        “gateway”: “value1”
        “network”: JSONObject {
                “nodeId”: “value2”
                “battery”: “value3”
                “time”: “value4”
        }
        “values”: JSONObject {
                “parent”: “value5”
                “temp”: “value6”
                “humid”: “value7”
                “accX”: “value8”
                “accY”: “value9”
                “accZ”: “value10”
        }
}
```

Once this message is sent, the information can be consulted online through the COSM website. Every time this request is sent, the values will be updated in the web application and the historical data stored.

Examples of data visualization are provided in section 4.4.

4.3.2 CSV

For his second example, the idea trying to reduce to the max the amount of data sent. This way, PUT method will be used, resulting in the following header.

```
/* HTTP client */
PUT /v2/feeds/FEED_ID
Host: www.api.cosm.com/v2
X-ApiKey: API_KEY
Content-Length: LENGTH_CSV_DATA
Content-Type: text/csv
```

As in the previous example, FEED_ID is the identifier provided by COSM for the creation of the new data feed, and API_KEY the secure sharing key created for the transaction.

The data is now translated into CSV format. The information sent in both examples is just the same. The only difference relies in the fact that with JSON information is nested into structures, and with CSV it is presented as raw values.

```
<gateway>,<nodeid>,<battery>,<time>,<parent>,<temp>,<humid>,<accx>,<accy>,<accz>
value1,value2,valu3,value4,value5,value6,value7,value8,value9,value10
```

The first row is the encoding/parsing template, needed to be stored both at node and server level. The second row is the real data sent over the link.

4.4 Results

With the experience obtained by the study and implementation of both JSON and CSV formats with real M2M devices, Gateway and M2M server, the conclusions are summarized in the following table:

Table 4-1. Summary of JSON and CSV

	Pros	Cons
CSV	• Compact format • Easy to implement	• No hierarchy on data • Encoder/parser needed at both ends
JSON	• Easily readable • Hierarchical data	• Less support in non-JAVA languages • More bandwidth needed

As one of the main EXALTED assumptions is that billions of M2M devices are envisaged to be present in the networks and they must be handled and monitored, bandwidth becomes a critical aspect.

This way, the final conclusion is that, if there are no unavoidable needs regarding structured data transmission, CSV format is the most suitable one (from the three alternatives listed, which are the most common options for monitoring tasks) for optimizing resource consumption of constrained devices such as the ones evaluated in EXALTED. It will imply developing extra functionalities at server side in order to correctly parse and handle the received information, but it will assure connectivity and scalability.

A comparison of the compression achieved by each format is presented in Figure 4.2. In this example, the amount of data sent is calculated for a simulated network composed by:

- 1 Gateway connected to the core network and handling connections for all end devices.
- 1000 end nodes transmitting 5 values in CSV format.

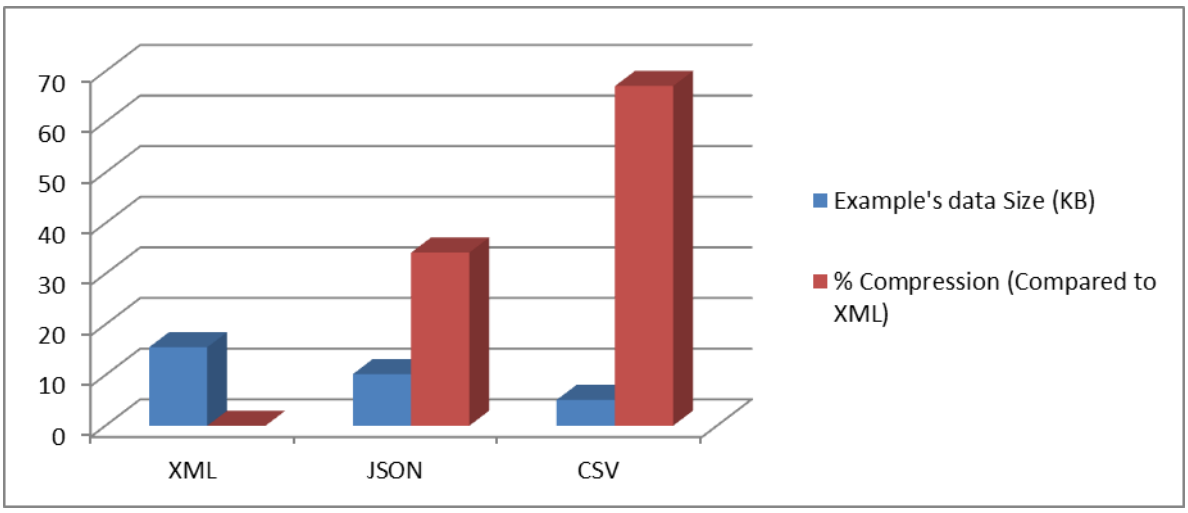

Figure 4.2. XML, JSON and CSV comparison.

The resultant size of the data sent in CSV is approximately half of the one related to JSON, and a third of the one sent with XML.

Finally, Figure 4.3 is a screen capture of the COSM web interface, in which the temperature values are pictured over a 24 hour period.

Any of the values sent (the ones described in sections 4.3.1 and 4.3.2) can be prompted and their historical data printed in a figure.

Raw messages can be also shown by clicking in the proper data format label (JSON, CSV or XML), depending on the model chosen.

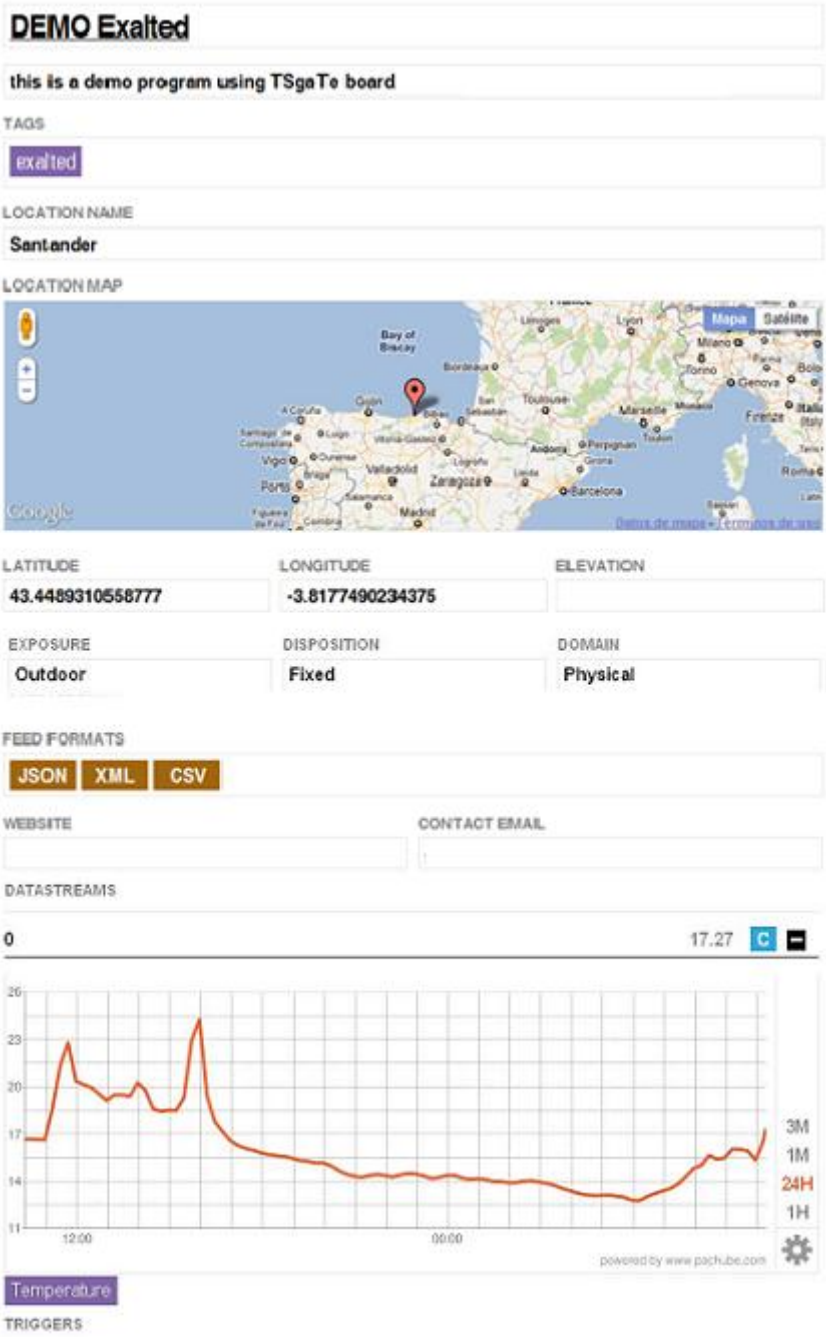

Figure 4.3. Visualization of data received in COSM.

5. Conclusions

Network monitoring is required to support the deployment of monitoring network services and elements by providing the necessary information from the current status of the network. It has mainly been used for fault detection and performance optimisation.

In this document, different types of network monitoring systems were introduced and the state of the art monitoring solutions for capillary networks (i.e. wireless sensor networks) was outlined. It was explained why these solutions are not fully compliant with EXALTED characteristics and requirements (i.e. reliable, scalable and configurable). The challenges and requirements for network monitoring were outlined and specific monitoring requirements for EXALTED were explored.

This document proposes a software architecture for network monitoring that has several advantages over the current solutions. Scalability is the main design consideration and it intends to support a large number of M2M devices. In addition, it introduces the standard interfaces allowing interoperability with other EXALTED architectures such as device management and self-diagnostics.

The proposed framework enables the intermediate devices (Gateways, Cluster Heads) to combine their monitoring status with the results from connected M2M devices to capture the status of the network. A hierarchical architecture tackles scalability by applying monitoring policies in different network levels to reduce monitoring network traffic load over the LTE/LTE-M network.

Finally, the lightweight monitoring mechanism implementation demonstrates that it is possible to build a way to post and retrieve efficiently data from M2M nodes without compromising their resources and overloading the network with unnecessary overheads. The reduction obtained is over 60% compared to traditional ways of monitoring non-M2M equipment, and the overhead introduces is minimum as the data is compressed at gateway level.

List of Acronyms

Acronym	Meaning
API	Application Programming Interface
CH	Cluster Head
CSV	Comma Separated Values
DL	Down Link
DM	Device Management
E2E	End-to-End
eNB	eNodeB
EXALTED	EXpAnding LTE for Devices
GW	Gateway
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
LTE	Long Term Evolution
LTE-M	LTE extension for M2M services
M2M	Machine-to-Machine
MIB	Management Information Base
MOI	Managed Objects Interface
NGS	NMM Global Status
NMM	Network Monitoring Module
NMS	Network Monitoring/management Server
OMA	Open Mobile Alliance
OMA-DM	OMA Device Management
SDM	Self-Diagnostic Module
SNMP	Simple Network Management Protocol
UFI	User Friendly Interface
UL	Up Link
UPnP	Universal Plug and Play
WP2	EXALTED Work Package 2 (use cases, business plan and technical requirements)
WP3	EXALTED Work Package 3 (LTE-M system)
WP4	EXALTED Work Package 4 (end-to-end M2M system)
WP5	EXALTED Work Package 5 (security and provisioning)
WP6	EXALTED Work Package 6 (device improvements)
WP7	EXALTED Work Package 7 (integration and proof of concepts)
WSN	Wireless Sensor (and actuator) Network
XML	Extensible Markup Language

References

[1] FP7 EXALTED WP2 D2.1 “Business Models, Use cases & Technical Requirements”

[2] M. Reza Akhavan, T. Watteyne, H. Aghvami, Enhancing the Performance of RPL Using A Receiver-Based MAC Protocol in Lossy WSNs, In 18th International Conference on Telecommunications (ICT 2011), Ayia Napa, Cyprus, May 2011.

[3] B. Zhang and G. Li. Analysis of network management protocols in wireless sensor network, In Proceedings of the International Conference on Multimedia and Information Technology, pp. 546-549, Los Alamitos, CA, USA, 2008.

[4] I. F. Akyildiz, and M. C. Vuran, Wireless Sensor Networks, John Wiley, 2010.

[5] G. Hoblos, M. Stroswiecki, and A. Aitouche. Optimal design of fault tolerant sensor networks, In Proceedings of IEEE International Conference on Control Applications, pp. 467- 472, Anchorage, AK, USA, September 2000.

[6] D. Nadig and S.S. Iyengar. A new architecture for distributed sensor integration, In Proceeding of IEEE Southeastcon’93, Charlotte, NC, USA, April 1993.

[7] C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor information networking architecture and applications, IEEE Personal Communications, 8(4):52-59, August 2001.

[8] K. Sohrabi, D. Minoli, T. Znati, Wireless Sensor Networks: Technology, Protocols, and Applications, John Wiley, 2007.

[9] W. L. Lee, A. Datta, and R. Cardell-Oliver, Network Management in Wireless Sensor Networks, Handbook on Mobile Ad Hoc, 2007.

[10] H. Song, D. Kim, K. Lee, and J. Sung, Upnp-Based Sensor Network Management Architecture, In Proceedings of ICMU Conference, April 2005.

[11] M. Turon, MOTE-VIEW: A Sensor Network Monitoring and Management Tool, In Proceedings of IEEE EmNetS-II Workshop, May 2005.

[12] G. Tolle and D. Culler, Design of an Application-Cooperative Management System for Wireless Sensor Networks, In Proceedings of EWSN’05, pp. 121-132, Istanbul, Turkey, January 2005.

[13] N. Ramanathan, E. Kohler, and D. Estrin, Towards a Debugging System for Sensor Networks, International Journal for Network Management, vol. 15, no. 4, pp. 223–234, 2005.

[14] J. Zhang, E.C. Kulasekere, K. Premaratne, and P.H. Bauer, Resource Management of Task Oriented Distributed Sensor Networks, In Proceedings of IEEE ICASSP Conference, May 2001.

[15] A. Boulis and M.B. Srivastava, Node-level Energy Management for Sensor Networks in the Presence of Multiple Applications, In Proceedings of IEEE PerCom Conference, Mar. 2003.

[16] N. Ramanathan and M. Yarvis, A Stream-oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications, In Proceedings of IEEE EmNetS-II Workshop, May 2005.

[17] C. Fok, G. Roman, and C. Lu, “Mobile Agent Middleware for Sensor Networks: An Application Case Study,” in Proc. IEEE ICDCS Conf., June 2005.

[18] A. Erdogan, E. Cayirci, and V. Coskun, “Sectoral Sweepers for Sensor Node Management and Location Estimation in Adhoc Sensor Networks,” in Proc. IEEE MILCOM Conf., Oct. 2003

[19] R. Tynan, D. Marsh, D. OKane, and G. M. P. OHare, “Agents for Wireless Sensor Network Power Management,” in Proc. IEEE ICPPW Conf., June 2005.

[20] L.B. Ruiz, J.M.S. Nogueira, and A.A.F. Loureiro. MANNA: a management architecture for wireless sensor networks. IEEE Communications Magazine, 41(2): 116–125, February 2003.

[21] B. Deb, S. Bhatnagar, and B. Nath, STREAM: Sensor Topology Retrieval at Multiple Resolutions, Kluwer Journal of Telecommunications Systems, vol. 26, no. 2, pp. 285–320, 2004.

[22] T.H. Kim and S. Hong, “Sensor Network Management Protocol for State-Driven Execution Environment”, In Proceedings of ICUC Conference, Oct. 2003

[23] FP7 EXALTED WP2 D2.3 “The EXALTED System Architecture”

[24] FP7 EXALTED WP5 D5.1 “Security and provisioning solutions”

[25] M. Ivester, A. Lim, Interactive and Extensible Framework for Execution and Monitoring of Wireless Sensor Networks, First International Conference on Communication System Software and Middleware, pp. 1-10, Jan. 2006.

[26] P. Levis, N. Lee, M. Welsh, and D. Culler, TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications, In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003).

[27] FP7 EXALTED WP2 D2.1 “Description of baseline reference systems, scenarios, technical requirements & evaluation methodology”

[28] RFC3411: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks.

[29] FP7 EXALTED WP4 D4.3 “Device Management”

[30] FP7 EXALTED WP6 D6.3 “Final specification of the reliable device implementation”

[31] H. S. Nwana, L. Lee, and N. Jennings, Coordination in Software Agent Systems, BT Technology Journal, 14(4), October 1996, pp.79-88.

[32] OMA-DM working group: http://www.openmobilealliance.org/Technical/DM.aspx

[33] TSmarT hardware platform, http://www.tst-sistemas.es/en/products-2/tsmote/

[34] FP7 EXALTED D4.4 “Traffic Aggregation”

[35] Cosm - Internet of Things Platform Connecting Devices and Apps for Real-Time Control and Data Storage, https://cosm.com/

[36] ThingSpeak free middleware platform for internet of things, https://www.thingspeak.com/

[37] DEXCell from DEXMA monitoring software, http://www.dexmatech.com/en/products/