Document No.	Security Classification	Date
/pSHIELD/D6.1	PU	31.01.2012

Project no: 100204

pSHIELD

pilot embedded **S**ystems arc**HI**tectur**E** for multi-**L**ayer **D**ependable solutions

Instrument type: Capability Project

Priority name: Embedded Systems / Rail Transportation Scenarios

Platform Development Report

For the pSHIELD-project

Deliverable D6.1

Partners contributed to the work:

HAI, Greece
ASTS, Italy
SESM, Italy
UNIGE, Italy
CS, Portugal
CWIN, Norway
ATHENA, Greece
THYIA, Slovenia
MAS, Norway

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2012)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Authors and Approvals

Authors		Date	Signature
Name	**Company**		
Nikos Pappas	HAI		
Mariana Esposito	ASTS		
Francesco Flammini	ASTS		
Alfio Pappalardo	ASTS		
Przemyslaw Osocha	SESM		
Lucio Marcenaro	UNIGE		
Jose Verissimo	CS		
Gareth May-Clement	CS		
Zahid Iqbal	CWIN		
Sarfraz Alam	CWIN		
Kyriakos Stefanidis	ATHENA		
Spase Drakul	THYIA		
Gordana Mijic	THYIA		
Ljiljana Mijic	THYIA		
Nastja Kuzmin	THYIA		
Josef Noll	MAS		
Reviewed by			
Name	**Company**		
Approved by			
Name	**Company**		

Modification History

Issue	Date	Description
Draft A	10.11.2011	First issue for comments, Table of Contents
Issue 1	01.12.2011	WSN platform by ASTS
Issue 2	10.01.2012	FPGA Node by SESM, Cognitive Radio Node by UNIGE
Issue 3	16.01.2012	Security Integration by CWIN, Middleware Prototype by CS
Issue 4	20.01.2012	Semantics, Policy Management and Hybrid Automata
Issue 5	25.01.2012	Encrypted Communications by Athena
Issue 6	26.01.2012	Nano, Micro/Personal Node (NMP) Prototype by THYIA
Issue 7	30.01.2012	Synergies across heterogeneous modules by MAS
Final	31.01.2012	pSHIELD Platform, Conclusions, Review by HAI

Contents

Figures

Tables

Glossary

AES	Advanced Encryption Standard
BSS	Basic Service Set
CCK	Complementary Code Keying
CCMP	Counter mode with Cipher block chaining MAC Protocol
CDMA	Code Division Multiple Access
CLA	Cross Layer Architecture
CPU	Central Processing Unit
CR	Cognitive Radio
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DDR3	Double-Data-Rate
DPRAM	Dual Ported RAM

DRA	Dynamic Resource Allocation
DRAM	Dynamic RAM
DSP	Digital Signal Processor
DSSS	Direct-Sequence Spread Spectrum
DVB-RCS	Digital Video Broadcasting Return Channel via Satellite
EAP	Extensible Authentication Protocol
EAP-TLS	EAP-Transport Layer Security
EAP-TTLS	EAP-Tunneled Transport Layer Security
ESD	Embedded System Device
ESs	Embedded Systems
FDMA	Frequency Division Multiple Access
FCC	Federal Communications Commission
FHSS	Frequency-Hopping Spread Spectrum
FPGA	Field-programmable Gate Array
FSK	Frequency-Shift Keying
I/O	Input/Output
JVM	Java Virtual Machine
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
NGN	Next Generation Network
OFDM	Orthogonal Frequency-Division Multiplexing
OS	Operating System
PBM	Policy Based Management
PC	Personal Computer
PEAP	Protected EAP
pS-ESD	pSHIELD Embedded System Device
pS-MS	pSHIELD Middleware Service
pS-OS	pSHIELD Overlay Service
pS-P	pSHIELD Subsystem
pS-P	pSHIELD Proxy
pS-SPD-ESD	SPD Embedded System Device
PSK	Phase-Shift Keying
QoS	Quality of Service
RAM	Random Access Memory
RF	Radio Frequency
SDR	Software Defined Radio
SDRAM	Synchronous Dynamic RAM
SIP	Session Initiation Protocol
SNMP	Simple Network Management Protocol
SPI	Serial Peripheral Interface
SOA	Service Oriented Architecture
SPD	Security Privacy Dependability
SSL	Secure Socket Layer
TDMA	Time Division Multiple Access
TKIP	Temporal Key Integrity Protocol
TPM	Trusted Platform Module
UART	Universal Asynchronous Receiver/Transmitter

UCS	Use case Scenario
USB	Universal Serial Bus
VPN	Virtual Private Network
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WPAN	Wireless Personal Area Network
WRAN	Wireless Regional Area Network

1 Executive Summary

D6.1 is a deliverable inside pSHIELD WP6, “Multi-Technology System Developments”. The deliverable aims, primarily, at concentrating and registering material that will be used to demonstrate the basic technological ideas of pSHIELD, proving the feasibility of the concept and setting the ground basis for future improvements, research and implementations. The components of this material cover a wide spectrum from software and hardware, representing different points of pSHIELD focus, ranging from simple sensing units and more complex physical nodes to abstract software entities as Middleware and simulations of real life application scenarios as freight trains monitoring. To this direction, partners’ work throughout technical work packages (WP3-WP5) will be listed hereafter, paving the way to the exploration of usability, synthesis and composability of the overall pSHIELD architecture. A vertical testbed with as many as possible component synergies and subsystem groups, will have the scope of interoperating pSHIELD SPD modules to address adequately selected SPD concerns. The resulting demonstration platform will be tested in the process of WP6, in the framework of the application scenario and validated against SPD metrics and requirements set during the theoretical foundation of the project.

This Page is intentionally left blank

2 Introduction

The main goal of pSHIELD is to ensure that security, privacy and dependability (SPD) in the context of integrated and interoperating heterogeneous services, applications, systems and devices. Systems and services must be robust in the sense that an acceptable level of services is available despite the occurrence of transient and permanent perturbations such as hardware faults, design faults, imprecise specifications, and accidental operational faults.

The current deliverable is an introductory step towards the implementation of a Demonstration Platform, constituted of prototypical demonstrators, the diversity and technical capacity of which will provide adequate proof of concept for the pSHIELD idea. This validation will turn up as the outcome of the work evolvement of the consortium. Starting from the foundation of theoretical framework and requirements, passing then to the definition of "key" concepts and application scenarios, to result in the development of software and hardware prototypes and their demonstration as a unified (up to the possible extent) platform. The scope is to test the functionality of the SPD composable modules and the efficiency with which they manage real-life service requests, as well as the SPD level they can achieve, as the latter is substantiated throughout the project's terms and definitions. In this way, not only pSHIELD concept will be solidified, but the perspective of future research will become clearer and grounded on a solid basis.

The document is structured with its core being the description of the prototypical demonstrators, preceded by a chapter providing a synoptic view on the demonstration platform and a brief reminder of the selected centric application scenario. The final sections are dedicated to the deduction of conclusions, also in terms of exploring the synthetic potentialities of the components in favour of their wider use in real world applications.

3 pSHIELD Platform Structure

This document will register the demonstrating environment and the complete network architecture that is used to conduct pSHIELD validation actions. This demonstration platform is based on local demonstration prototypes of all partners that will participate in WP6, either by hardware or software means, according also to the revised pSHIELD Technical Annex. The prototypes (indoor/outdoor) are composed of sensors, ranging in processing power and technical capabilities accompanied or represented of software modules that will demonstrate the project's basic notions and key concepts. The synthesis of the demo platform should serve efficiently a testing procedure comprised of functional tests, performance tests, traffic measurements, security evaluation tests, trace analysis, confrontation of hazardous situations and emulation of the conditions that are encountered in the real world deployments of applications. This document has the main objective to initially provide accurate descriptions of the partners' local demo sub-systems and subsequently prepare the way for wider demonstrations, intra-system cooperation, composability and a range of applications.

The list of demonstrating prototypes along with the basic pSHIELD ideas they represent includes the following:

- **Monitoring Trains with WSNs**
  - ✓SPD functions in an integrated embedded sensor testbed
  - ✓SPD metrics based composability
- **Nano, Micro/Personal Node Prototype**

✓Security interworking between embedded sensors and Telecom service platform SPD metrics based composability

- **FPGA Power Node Prototype**
  - ✓Modular system reconfiguration
  - ✓Self-dependability at node layer
  - ✓Hardware and software security and privacy service provider
  - ✓Management of power sources
- **Cognitive Radio Node Prototype**
  - ✓Reconfigurable radio components with waveform Tx parameters
  - ✓Sensing mechanisms to acquire awareness about resources
  - ✓Cognitive algorithms elaborating available resources
  - ✓Embedded platform adaptation for validation of algorithms
- **Semantic Model Prototype**
  - ✓Basic functionalities of pSHIELD Middleware
- **Middleware Prototype**
  - ✓Demonstrating SPD driven composability
- **Policy based Management and Hybrid Automata model**
  - ✓Policy based approach and control laws for SPD composability
- **Security Integration across Heterogeneous Platforms**
  - ✓Sensor as a service approach
- **Platform for heterogeneous Wireless Sensor networks**
  - ✓WSNs from the pSHIELD perspective

The figure below is depicting several components in what could be an example (for illustrating purposes only) of a pSHIELD abstraction:

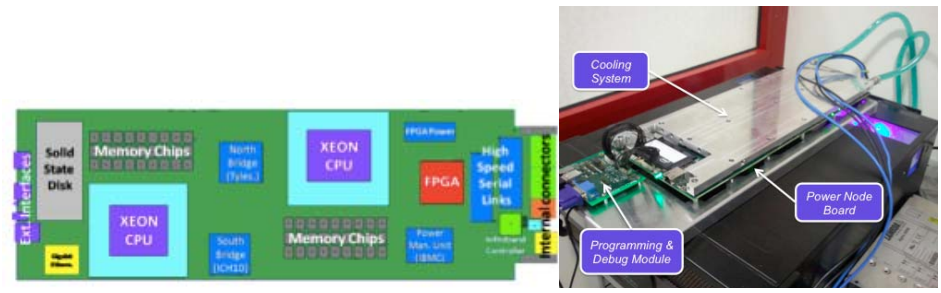

Figure 1 – pSHIELD Demo Components (arbitrary selection)

4 Application Scenario: Monitoring Trains

The use case of reference for pSHIELD project is the monitoring of freight trains transporting hazardous material. Therefore, the detection of abnormal operating or environmental conditions on board of vehicles as well as threats of burglary represents an example application of great interest for the freight train monitoring. The main objectives of this application scenario are to validate the technical concepts of SHIELD Security, Privacy and Dependability as a whole. In particular, in this use case, the following requirement has to be fulfilled:

- Secure handling of the critical information of the hazardous material;
- Secure and dependable monitoring of transport.

The overall monitoring system is highly heterogeneous in terms not only of detection technologies but also of embedded computing power and communication facilities. The ESs can differ in their inner hardware-software architecture and thus in the capacity of providing information security, privacy and dependability.

As mentioned above, the monitoring is aimed to the detection of abnormal operating or environmental conditions on board of vehicles as well as threats of burglary.

In the described context, several problems and challenges are to be solved. In particular, they are referred to:

- Availability: information (measured data) must be provided continuously according to soft or hard real-time constraints.

- Integrity: sensed, stored or transmitted data must not be corrupted accidentally or in a malicious way.
- Privacy: information must be accessed only by authorized users, for confidentiality reasons.
- Integration/expansion dependability: whenever any new ES needs to be integrated into the system, there should not be the need to develop a new protocol and/or driver and it should be also possible to easily evaluate the impact of the modification on the overall system dependability.

5 Nano, Micro/Personal Node (NMP) Prototype

5.1 Main features of NMP node prototype

There are three kinds of pSHIELD Node: Nano nodes, Micro/Personal (NMP) nodes and Power nodes. Nano nodes are typically small ESD with limited hardware and software resources, such as wireless sensors. Micro/Personal nodes are richer in terms of hardware and software resources, network access capabilities, mobility, interfaces, sensing capabilities, etc. Power nodes offer high performance computing in one self-contained board offering data storage, networking, memory and multi processing. While the three pSHIELD Node types cover a variety of different ESDs, offering different functionalities and SPD capabilities, they share the same conceptual model, enabling the pSHIELD seamless Composability.

5.1.1 SPD and legacy nodes in pSHIELD

The technology advancements in computing hardware and software enables a new generation of small ESDs to perform complex computing tasks. Extremely small sensor devices provide advanced sensing and networking capabilities. In parallel, many operating systems targeting these types of devices have been developed to increase their performance. The way for designing pSHIELD NMP Nodes is twofold:

1. To design completely new NMP nodes that are compliant with the pSHIELD system design.
2. To keep legacy technologies as they are, developed for many applications including those that are targeted in pSHIELD, which assumes that a heterogeneous infrastructure of networked ESDs like IEEE 802.15.4, IEEE 802.11, etc. An ordinary sensor technology (not all, since we need those that are designed for ES) permits to consider an augmentation of SPD functionalities at different levels of the hardware and firmware modules. This means an enhanced nano, micro/personal node with physical layer and protocol stack composed of existing and new SPD technologies. As result of this integration new types of networked SPD ESDs will be created. This new SPD ESDs will compose a heterogeneous SPD network infrastructure too.

Developing a NMP node equipped with some Legacy functionalities and with the pSHIELD Node Adapter (pSNA), we obtain a composable pSHIELD node. It means that it has all desired SPD functionalities and services for the pSHIELD application scenario selected. Additionally to that, the NMP node keeps almost all desired functionalities of a standardized sensor technology with additional SPD features that make it composable into the pSHIELD system architecture.

5.1.2 NMP node operating systems

Selection of the operating system (OS) for the demonstrator is an important design constraint, since we need to decide in which sensor prototype platform will be realized SPD functionalities. The only

requirement that we posed for this operating system is related to its possibility to be designed for embedded devices. There are two candidates for that: TinyOS and Contiki.

5.1.2.1 TinyOS

This operating system (OS) is a free and open source operating system and platform that is designed for WSNs. It is an embedded operating system, written in the nesC Programming language as a set of cooperating tasks and processes. NesC is actually a dialect of the C programming language that is optimised for the memory limitation of the sensor networks.

5.1.2.2 Contiki Operating System

Contiki is also an open source, highly portable, multi-tasking operating system for memory-efficient networked ESDs and WSNs. It is mainly designed for a microcontroller with small amount of memory. The key advantage of Contiki OS is its IP communications (both IPv4 and IPv6). It is flexible for a choice between full IP networking and low-power radio communication mechanisms. Contiki is written in the C programming language and consists of an event-driven kernel, on top of which application programs can be dynamically loaded and unloaded at run time. Contiki has been ported to different hardware platforms, such as MSP430, AVR, HC 12, and Z80.

5.1.3 Wireless Sensor Networks composed of pSHIELD nodes

The pSHIELD network architecture for the railway application scenario, the concept of four functional layers with SPD functionalities and core services is a homogenous network as in Figure 2.2 of the Technical Annex. By introducing more applicational scenarios as in nSHIELD and Legacy ES nodes and Legacy ES Networks, the final architecture becomes a hybrid heterogeneous network (HHN). Heterogeneous in the sense of coexistence of different technologies (IEEE 802.15.4, IEEE 802.11, UMTS, etc.) and hybrid in the sense of a network between central and pure decentralised architecture. The figure below illustrates a WSN composed of Nano, Micro/Personal and Power Node which can be used also as a Gateway.

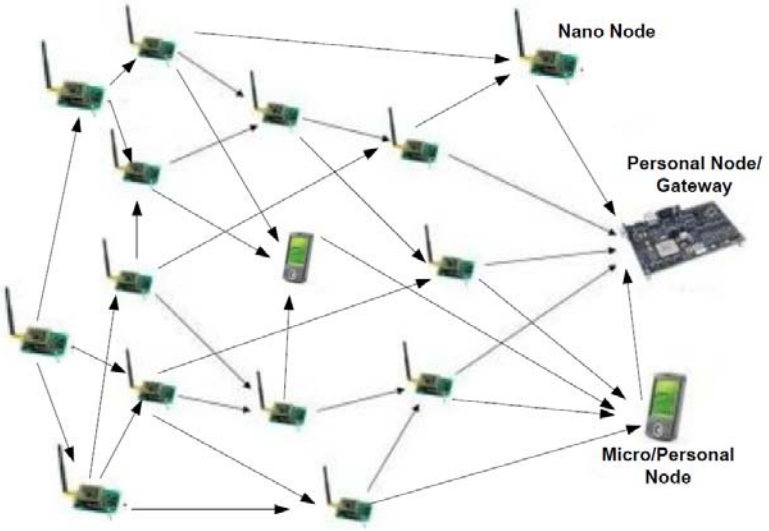

Figure 2 - WSN composed of NMP and power nodes

For example, in the pSHIELD network it can be designed to track wagons that pass through certain geographical areas up to the destination. Therefore, the network may switch between being a monitoring network (inside the wagons, or trains) and a data collection network (outside, railway road or railway track). During the long periods of inactivity when no monitored wagons are present, the network will simply perform the monitoring function. Each NMP or power node will monitor its sensors waiting to detect an alarm. Once an alarm event is detected, all or part of the network, will switch into a data collection network and periodically report sensor readings up to a GW that tracks the wagon. Due to this multi-modal network behaviour, it is important to develop a single architecture that can handle these application scenarios as well as other scenarios.

5.1.4 Middleware

The sensor node for WSNs differs so much in terms of HW platforms. The recent development of sensor node middleware is showing that we have quite a large number of middleware for WSNs. Most of the middleware we have studied are built on top of TinyOS. There are other OSs like Contiki, Mantis, SOS, and t-kernel. It is important to note that the scope of middleware for WSN is not restricted to the sensor network alone, but also covers external networks connected to the WSN (such as Internet) as well as the applications interested in querying sensor data through such external network. Standards such as 6LoWPAN (which used IEEE 802.15.4) and Web Services running directly on the sensor node allow integrating them into the Internet of Things (IoT). However, nodes which are capable to run the internet stack directly are either very expensive or not very energy-efficient. There have been several efforts to implement the Internet Protocol Stack on small constrained devices. The LoWPAN and 6LoWPAN protocols try to port the IPv4 and IPv6 Protocols on small devices. This enables running services on the application layer directly on sensor nodes. The Web service technology is often used to connect and access sensors and actuators through the Internet. The recent middleware approaches use different technique. For example, such middleware are Sensorpedia (Web 2.0 based), TinyDB (Database oriented), Mate (Virtual Machine based), Agilla (Mobile Agent), TinyLime (tuple space) and TinyCubus (cross-layered). Taking in consideration that pSHIELD SPD network is composed of SPD and Legacy Nodes it is obvious that we have a complex HHN structure where the standard OSI layers are defining the overall network requirements in sense of the HW & SW components. On the physical layer (PHY) different NMP nodes will coexist in the same pSHIELD network. Above PHY different protocol stacks for different Legacy NMP nodes are increasing the complexity of the overall pSHIELD network design. Hydra platform is a new concept that is realized in such a way that between physical and application layer is a middleware. The main goal was to develop a middleware that is 'inclusive' which means that it will be possible to enable any device to be detectable and usable from a Hydra application. The concept is based on the work of Rozanski and Woods, and the Hydra architectural descriptions are in line with the IEEE 1471 standard. For the NMP prototype platform design concept we will explain in the following section how it can be composed by an operating system, middleware and the application layer. The Hydra middleware as in the figure below is a core technology that has a transparent communication layer, equally supporting centralized and distributed architectures.

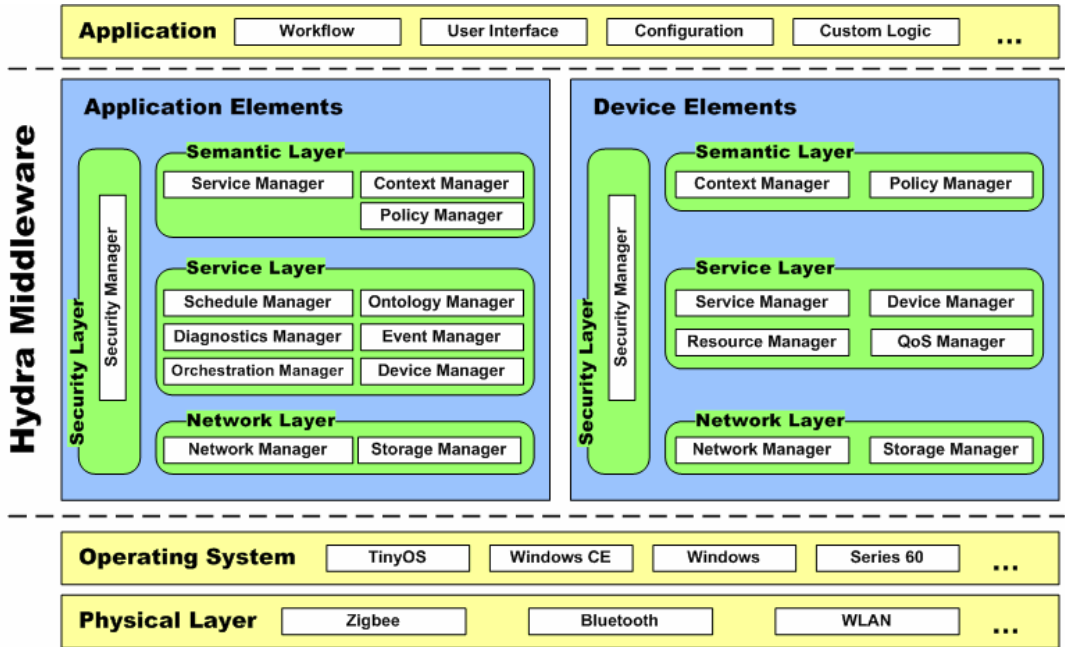

Figure 3 - The Hydra middleware layer

The biggest advantage of the Hydra middleware relies on the fact that allows developers to incorporate heterogeneous ESDs into their applications. This middleware can be incorporated in new and existing networks of distributed ESDs, which operate with limited resources: computing power, energy and memory. Additionally, Hydra-middleware provides easy-to-use web service interfaces for controlling any type of physical device irrespective of its network interface technology. Additionally, this middleware is based on a semantic Model Driven Architecture for easy programming and incorporate service discovery, P2P communications and diagnostic. In Hydra framework any physical devices, sensor, actuators or subsystem can be considered as a unique web service.

5.1.5 Multidimensional metric space

The SPD metrics is defined in D2.1.1 and D2.2.1 with key security & dependability attributes: availability, reliability, safety, confidentiality, integrity and maintainability and the system performance metrics that are important for WSN applications such as computational time, memory size, energy consumption and cost. Additionally, authenticity attribute is very important for WSN.

The key metrics for wireless sensor networks are grouped for SPD functions

- security
- privacy
- dependability

and basic functions:

- lifetime
- coverage
- cost and deployment
- response time

- temporal accuracy
- effective sample rate

These functions can be considered with the key aspects: threats, attributes and means in the main concept taxonomies: security, dependability, fault-tolerance, reliability and survivability (see D2.2.1). The importance of these functions is briefly discussed below. Many of these evaluation metrics are interrelated. For example, it may be necessary to decrease performance in one metric, such as sample rate, in order to increase another, such as lifetime. Taken together, this set of metrics form a multidimensional metric space (MMS) that can be used to describe the capabilities of a WSN and its nodes. The SPD capabilities of a prototype platform are represented by this MMS. A specific application deployment can be represented by a subset in this MMS. A system prototype platform can successfully perform the application if and only if the application requirements subset lies inside the capability of MMS.

5.2 NMP Node: Prototypes

This section provides some key details for the NMP sensor (NMPS) node prototypes with small energy-constrained sensor nodes that form a WSN. Development platform will be designed in such a way to facilitate security enhancements discussed in D3.2. (Chapters 4, 5, 6 and 7). For the application scenario, i.e., rail transportation of dangerous materials the best suited proof of the concept prototype is capability of a NMPS node to maintain information integrity, confidentiality, authenticity and system integrity by using symmetric or asymmetric key cryptography. Therefore, our SPD goal for the NMPS node prototypes is to take in consideration the following design constraints:

I. For RT scenario, which belongs also to critical infrastructure, high security of WSNs composed of secured NMPS nodes is compulsory

II. NMPS nodes are energy and resources-constrained

III. Secure ES firmware, secure boot, secure upgrade mechanisms, and TCG technologies are needed for enhancing security

5.2.1 Development Platform

Development platform has two separate prototypes:

1. NMPS node platform
2. TPM platform

The choice of the processor and memory performance is very important since the program memory sized defies performance (MIPs) and computational time (ms). Selection of all other components for both platforms is constrained with constrains I, II and III.

5.2.2 NMPS node prototype

Before we decided which type of tiny sensor node will well suited with the pSHIELD requirements we investigated many suitable solutions. Fig illustrates the most recent sensor platforms that can be used for NMPS node (generic sensing type or gateway). For video applications the current sensor node platforms are showing lack of processing power and memory sizes. Therefore, low-resolution image sensors are considered for NMPS node. Additional goals for the NMPS node are:

- The node should have the memory-performance size 100-1000 KB and 10-100 MIPS
- WSNs will multi-tier type. For example Tier “0” is has nano nodes, tier “1” micro/personal nodes, and tier “2” more powerful micro/personal nodes as gateways, and tier “3” has power nodes
- The NMPS nodes should be able to connect: low-resolution camera, passive infrared (PIR), acoustic/ultrasound, temperature, pressure, humidity, etc.
- It should have sufficient low power consumption when is used with a battery
- It should allow a wide range of applications
- USB interface for programming the applications and data retrieval
- Separate USB interface will be for radio module
- To connect the image sensor and other sensors an expansion connector is used

5.2.2.1 Microcontroller/Microprocessor

First of all, choose of a microcontroller unit (MCU) based on several requirements such as low power consumption, rich on-chip peripherals, RAM and ROM, etc. The table below shows the comparison of the MCUs for three different types of nodes.

MCU	RAM (kB)	FLASH (kB)	Active (mA)	Sleep (µA)	Sensor Nodes
Atmega644/V (Atmel)	4	64	0.4	0.1	Nano
AT91SAM7128 (Atmel)	32	128	30	10	Micro
STM32W108B* STMicroelectronics	8	128	6@12MHz	<1	pSHIELD NMPS node

Table 1 - MCU comparison

(*) STM32W chip has integrated IEEE 802.15.4 radio at 2.4.GHz

The table above illustrates that Atmega644P/V has the lowest consumption for both active and sleep modes. It is a good candidate for nano node. The operating voltage is 1.8V. It uses an advanced RISC architecture where most of the 131 instructions only require one clock cycle to be executed and up to 20 Million Instructions per Second (MIPS) at 20MHz. It also provides all the basic peripherals for microcontroller with additional USART port, Timer and PWM modes. 4kB RAM is smaller compared to 10kB RAM (MSP430F16x). Although flash sizes are useful for large application programs, they are not the limiting factor in developing WSN applications. AT91SAM7S128 is a member of a series of low pin count Flash microcontrollers based on the 32-bit ARM RISC processor that runs at up to 55 MHz, providing 0.9 MIPS/MHz. It features a 128 Kbyte high-speed Flash and a 32 Kbyte SRAM, a large set of peripherals, including a USB 2.0 device and a complete set of system functions minimizing the number of external components. STM32W108 family is an excellent candidate for NMPS node since it has 32-bit ARM Cortex-M3 core running at 24MHz, considerably high RAM and FLASH memory with low power

consumption and an integrated IEEE 802.15.4 radio at 2.4 GHz! It will be furthermore investigated in the following up project nSHIELD, which is a three year project.

5.2.2.2 AT91SAM7S NMPS node prototype

The figure that follows illustrates the prototype architecture that we investigated. The expansion interface unit is used to connect and evaluated different NMPS node elements like sensors, TPM modules, etc. For example we investigated the image sensors: Agilent ADMC-1670 CIF, ADNS-3060 concurrently by using two independent UARTs and a shared SPI bus. In addition a reference camera from CoMedia C328R with VGA resolution from 80x60 to 640x480 is examined.

AT91SAM7S family offers RAM size of 8 – 64 kB and FLASH memory 32 – 256 kB. For an application if more RAM is necessary, a FRAM memory chip can be used. This is limited to 32 kB, but offer unlimited write/erase cycles on no wait states when writing. For example, if a 2MB FLASH device was specified for 100.000 write/erase cycles with one 100 kB frame written every 10 seconds, the devices would be expected to fail after ~230 days.

5.2.2.3 Design and i of a trusted NMPS node mplementation

TPM unit

The pSHIELD project aims to include trusted features in the sensor node design. We proposed TPM modules to enhance security of the devices.

Functionality of TPM

Details on ECC and RSA cryptography comparisons for energy-constrained NMP nodes are provided in D3.2. There are also important details on SW-TPM implementation

The most relevant functionalities of TPM for WSNs are:

- Cryptography operation engine (COE):

  The cryptography operation can be made by RSA or ECC for signature generation and message decryption, SHA engine and RNG. Every TPM is programmed with unique RSA or ECC key pair. The private part never leaves non-volatile storage area of TPM. When a nod is captured by an attacker the private part key would not be available to the attacker.

- Platform configuration register (PCR):

  TPM has a number (16) of PCR. The content stored in each PCR is a digest of messages in regard to the platform environment. PCRs are located in the non-volatile storage area and hence cannot be tempered with.

For example, if RSA is used the symmetric keys are typically generated by RNG. If an attacker can extract the initial random symmetric key, then it is possible for the attacker to compute all past and future random symmetric keys. Therefore, a strong RNG is very important for the effectiveness of symmetric key operations. RNG may be complaint with FIPS 104-2. The primitive random integer value is 2 . SHA-1 is used for TPM commands. It produces collision-free 20-byte hashed digest regardless of the input message. SHA-1 is frequently used because it is one-way function which is computationally infeasible to invert and used to develop the HMAC (Hashed Message Authentication Code). HMAC is an extension of SHA-1. Because the shared secret is not available to third parties an attacker cannot replay the

intercepted message due to the antireplay nonce or forge a valid HMAC for tempered message to circumvent the HMAC check. PCR value can be preserved even the TPM is turned off.

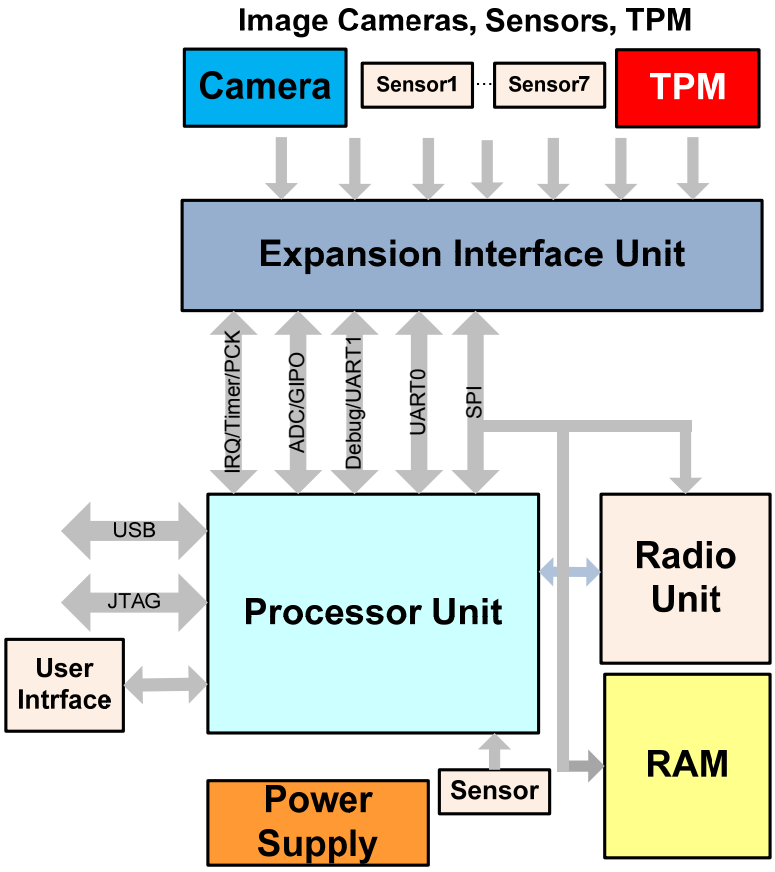

Figure 4 - NMPS node prototype architecture

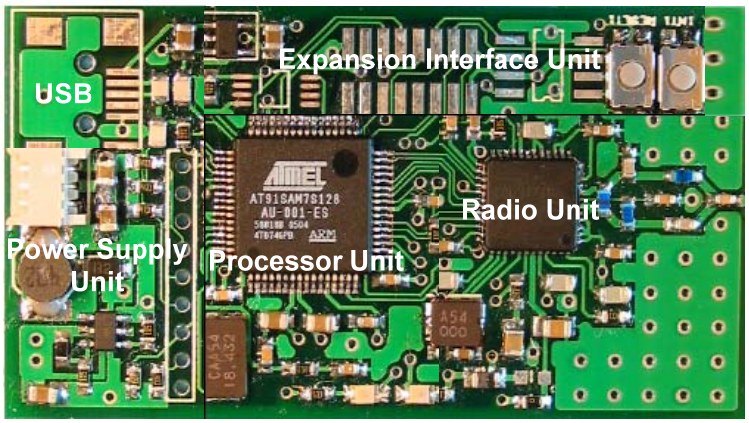

Figure 5 - A demo development prototype board

The way forward was to design a separate TM module. The contribution for trusted NMPS node of this deliverables includes the following:

1. Design of a trusted TPM platform for NMPS nodes, which include standard TPM chip. It is needed for cryptography (e.g., PKC, or ECC) and remote attestation in WSNs.
2. Extensive evaluation of trusted TPM unit in terms of cryptography algorithms, computation time, power consumption, cost, etc.
3. A proof-of-concept to use such trusted NMPS node for different applications where security enhancements are required (key management, secure SW update, secure remote attestation, etc).

The objective of a TPM is to provide a hardware-based root of trust for a device. For example, TPM has

- Cryptography operation engine (COE
  - TPM is programmed with a unique RSA key pair and the private part never leaves nonvolatile protected memory
  - RSA engine for signature generation and message decryption
  - Secure Hash Algorithm (SHA) Engine
  - Random Number Generation (RNG)
- Platform Configuration Register (PCR)
  - Stores integrity-sensitive messages in regard to platform environment
  - Located in nonvolatile protected memory (temper-proof)

For the microcontroller we selected AT91SAM7128 and for TPM Atmel AT97SC3203S[1] illustrated in the figures below (block diagram and demo board). It is a fully integrated security module for embedded systems. TPM unit is connected through extension unit by using I2C interface.

[1] http://www.atmel.com/dyn/resources/prod_documents/5132s.pdf

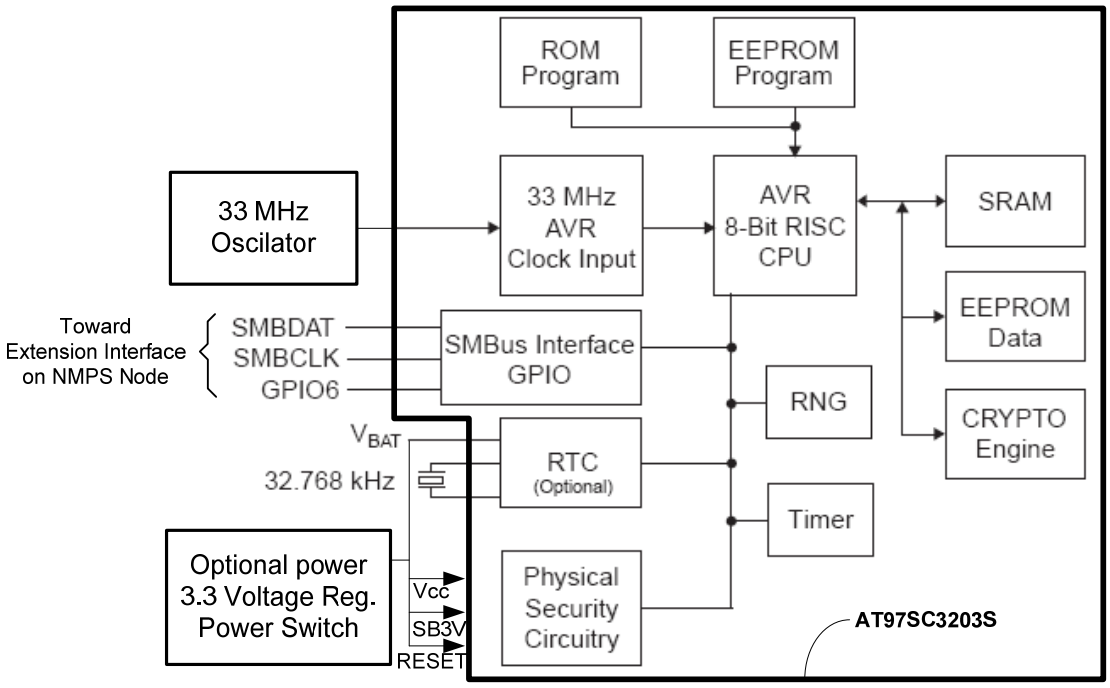

Figure 6 - Block diagram of Amtel AT97SC3203S TPM

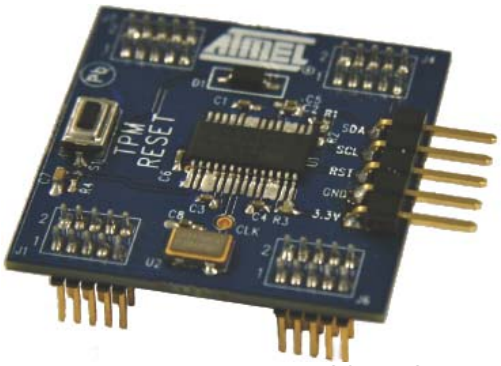

Figure 7 - TPM Amtel AT97SC3203S unit

A TMM is a micro-controller with additional security features that can store and protect sensitive information used to authenticate a trusted platform, e.g., passwords or cryptographic key. In contrast to a smart card, a TPM is usually logically (and physically) linked to a device or platform, not to a person, and provides additional secure hardware components. The figure above illustrates TPM unit. Atmel AT97SC3203S TPM chip size is 6.1 x 9.7 mm and costs 4.5 $ (large quantity. It can be easily integrated into NMPS node. With this we will achieve a compact NMPS node solution for many different applications.

5.2.2.4 SW-TPM implementation

Their experiments indicate that this optimization can significantly reduce SW-TPM overheads (an average of 6.51X execution time reduction and 6.75X energy consumption reduction for individual TPM commands, and an average of 10.25X execution time reduction and 10.75X energy consumption

reduction for applications). This work demonstrates that ECC-based SW-TPMs are a viable approach to realizing the benefits of trusted computing in resource-constrained embedded systems.

The TPM security features are very useful in many embedded systems. Some embedded systems cannot be augmented with a conventional TPM chip because of the area and cost constraints. Here, the feasibility of a SW-TPM is explored, which performs the same functions as a hardware TPM, i.e., supports all the three roots of trust, as well as other cryptographic capabilities. SW-TPM does not provide the same security level as a TPM chip. Executing the SW-TPM in a protected execution domain of the CPU (e.g., ARM Trust-Zone), and using on-chip memory, provides resistance to software attacks, including compromises of the OS, and a limited number of physical attacks (see D3.2 for further details).

The implementation of SW-TPM is adapted from the public domain TPM emulator, which provides basic TPM functions, such as RSA cryptography and HMAC and SHA-1 hashing functions, and provides several TPM commands. The emulator has been changed as follows:

- **Random number generation**: A hash-complemented Mersenne Twister (MT) random number generator is used, *i.e.,* we run the output of MT through SHA-1
- **ECC**: SW-TPM supports ECC in the binary field GF(2^m). ECC on this embedded platform is used because of its small key sizes compared to RSA for offering the same security robustness. Hence, it requires less resources such as processor cycles and energy. ECC-enabled SW-TPM supports key generation and validation, digital signature generation and verification, encryption, and decryption. Supported ECC key sizes are 224 bits (equivalent to 2048-bit RSA keys), 192 bits (not equivalent to RSA key), and 160 bits (equivalent to 1024-bit RSA keys)
- **AES_CBC cryptography**: SW-TPM supports the Advanced Encryption Standard (AES) algorithm, running in Cipher Block Chaining (CBC) mode. This engine is specifically used for ECC encryption and decryption, and for decrypting AIK credentials

5.2.3 Experiments

It is known that symmetric key cryptography consumes less energy than RSA (asymmetry keys, encryption key is different from decryption key). For example XTEA encryption consume approximately 10 times less energy compared to HW RSA encryption, and approximately 12.000 times less energy compared to SW RSA encryption. A strategy to adopt will be for nano nodes to use symmetric cryptography, and for critical applications like the pSHIELD scenario, asymmetric cryptography should be used. For example, asymmetric cryptography can be used to exchange a new symmetric key daily or hourly (this is called rekey process). An application can select to store the session keys in TPM.

An NMPS node A requests a new symmetric key from a NMPS node B, i.e., Gateway (GW). Node A initiates this process hourly or daily by generating a random number Na (nonce) and encrypts the nonce along with the request (Req) command using GW's public key (Pkgw) before transmitting it to the GW. The purpose of nonce is to defend against reply attacks. After receiving Req message from Node A, the GW decrypts the message with its private key Sgw. The GW responds to the Req command by generating a new symmetric session key Kba and encrypts it together with Na using a public key Pka before transmitting it to node A. Node A decrypts the message the from the GW with it its private key Ska and obtains the new symmetric key Kba. Nose A and the GW can then use Kba for future communications as in the figure below. Therefore, link level secure communications can be achieved by passing the returned cipher over the radio. In the case that the key are stored in Ram or EEPROM it is not secure, because that the information can be extracted from EEPROM and RAM in 1 min. Therefore, storing the key in TPM chip for these infrequent operations is more safely.

Group key establishing can be achieved by combination of sensor node symmetric session key request operation and sensor node symmetric session key assignment operation. For example if node A wants to communicate with node B and C, node A will request a new group session key from the GW via the

session key request operation. After receiving the key request operation from Node A, the GW generates a new symmetric key Kabc. The GW assigns Kabc to the Node B and C via two session key assignment operations before transmitting Kabc to Node A. Finally, Node A, B and C we perform secure communications using the group session key Kabc.

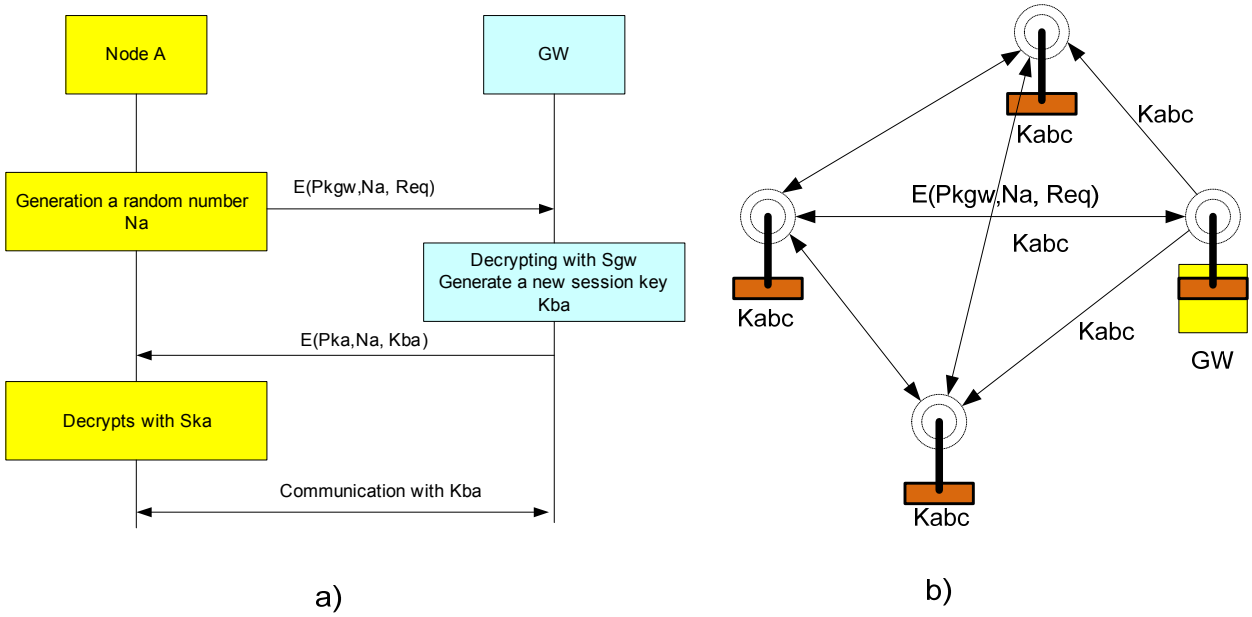

Figure 8 - Symmetric session key request operation with trusted NMPS nodes

5.2.4 Future Work

Integrity of a node can be verified by an attestation protocol, which used TPM as was proposed in the previous section. By to enable a WSN operator to react to tempering attempts, information about the node integrity needs to be exchanged through the network. If such information is exchanged overtly, attacker may be aware of the fact that the network is being monitored. Analyses of exchanged information may even reveal how often such information is exchanged and if no appropriate cryptography countermeasures are taken, it may also be possible to tell what information is exchanged. The problem is that every integrity protocol needs a secure channel between devices. Recently was proposed a “covert channel” for hidden transportation of integrity monitoring messages. The current work presented in this deliverables will be extended toward a new approach for enhancing security regarding the system integrity.

6 FPGA Power Node Prototype

In order to demonstrate the capabilities of the proposed pSHIELD SPD Power Node Layer Architecture, a case study has been implemented using the base architecture as described in D3.3. The demonstrator is coherent with the reference use case of pSHIELD project, based on the monitoring of freight trains transporting hazardous material. Namely, it consists of detecting the intrusion on different cars of the freight train, and raising alarms to a control center.

This Power Node is implemented on an FPGA, with the capability of runtime reconfiguration.

The current use case demonstrated some capabilities of a SPD Power Node, namely:

- **Dependability**, by detecting errors and tolerating them, through FPGA partial reconfiguration. After a fault being injected in the FPGA, affecting the demodulator, an error is detected and the FPGA is partially reprogrammed
- **Security**, by receiving encrypted data and being able to decrypt it
- **Self-Reconfiguration**, by reconfiguring the FPGA for adapting to a new function in run-time
- **Metrics**, by collecting and providing data such as the number of messages received, errors detected, etc.
- **Composability**, by providing discovery and composability information, such as the identification of the modules and its characteristics, that build-up the SPD Power Node
- **High performance**, by performing some intensive processing in real-time
- **Legacy** component **integration** in pSHIELD, by providing SPD functionalities to legacy components

6.1 Context

The scenario consists on the use of FSK modulation to transmit data between intrusion detection sensors placed in different cars of a freight train, to an SPD Power Node, which in turn processes the signals and send information to a control center through the pSHIELD network.

The intrusion detection systems are embedded devices which include a remote proximity sensor and a data encryptor. The remote proximity sensor is continuously measuring a distance to a nearby object. The encrypted data is then modulated, using FSK modulation, and transmitted to the Power Node. Each device modulates the signal with a different carrier, so that the Power Node is able to receive signals from different sources, assuring redundant sensors.

The Power Node receives the signals, demodulates them, decrypts, processes the data and sends to a control center through the pSHIELD Network.

The Control Center is a remote device, which could be a personal computer, tablet or mobile phone equipped with a web browser, able to visualize data and act upon.

The figure below presents the demonstrator context: the SPD Power Node is located in a central car of the freight train. It receives FSK modulated and encrypted data from other cars and delivers the plain information to the Control Center through the pSHIELD network.

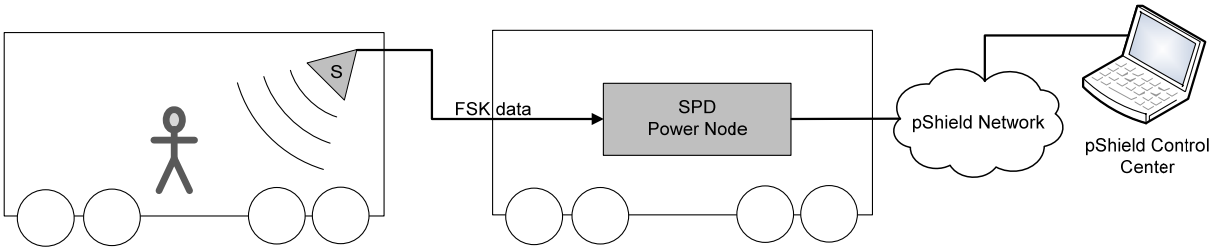

Figure 9 - Power Node demonstrator context

In our demonstrator we shall use, however, a single sensor using two distinct carriers, emulating sensor redundancy.

6.2 Demonstrator Power Node Architecture

The full demonstrator context is presented in the figure below. It consists of two different systems: the intrusion detector and the FSK Demodulator SPD node. The first receives data from an intrusion data generator, and is connected to a push button from where it is possible to request it to switch between the two carriers that are used for FSK modulation. This system then sends encrypted and FSK modulated data to the second system, the FSK Demodulator SPD Node. This system is also connected to a push button, to inject an internal fault into the Node. Finally, the FSK Demodulator Node is connected through Ethernet to the pSHIELD Network, from where a Control Center can receive data and control this node.

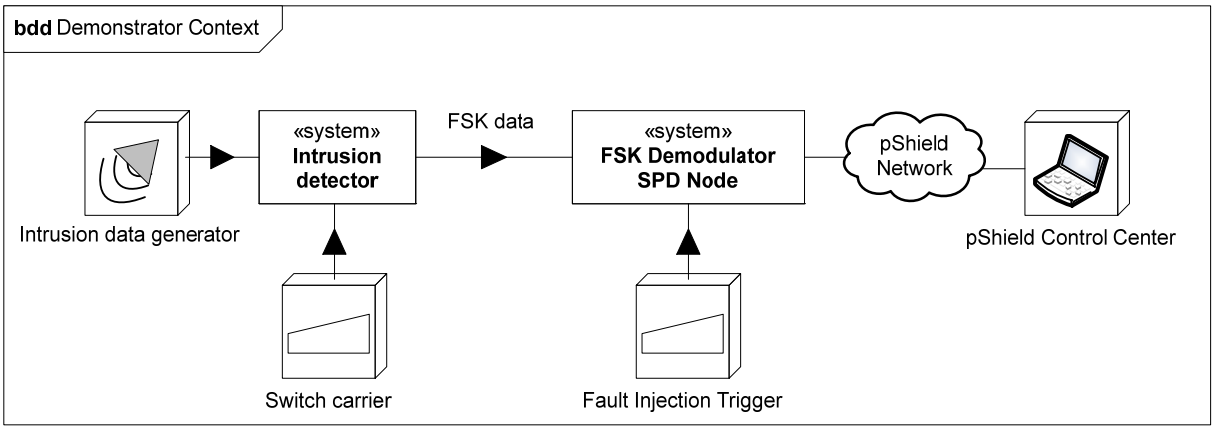

Figure 10 - Demonstrator context - block definition diagram

6.2.1 Intrusion Detector

The Intrusion Detector is implemented on the EP3C120F780 Cyclone III Altera FPGA.

It is composed of three basic blocks, as we can see from figure that follows:

- a **proximity sensor**, consisting on an intrusion data generator, which is based on a data file with emulated distances to the nearest object. No real sensor is being used. Reducing some level of complexity in a module that is not the main part of the current study. The values from this data file are periodically retrieved
- a **data encryptor**, encrypting the sensor data. This encryption is based on a Blowfish algorithm
- an **FSK modulator**, consisting of a hardware module (IP Core programmed on the FPGA), and using one of two predefined carriers

Intrusion data consists of:

- a data file containing values that represent the distances to the closest obstacle detected by the proximity sensor. The file is a text file in csv format. The structure follows:

  <progressive # of the sample>,<distance of the obstacle in cm>

  Each line will be a sample and each line will be sampled once per second
- a data encryptor uses blowfish algorithm with 64 length fixed key

- the FSK modulator works at 2 possible carrier frequencies of 1 kHz and 2 kHz. When the carrier is 1 kHz then the “Space” frequency is 968 Hz and the “Mark” frequency is 1031 Hz. While when the carrier is 2 kHz then the “Space” frequency is 1937 Hz and the “Mark” frequency is 2062 Hz

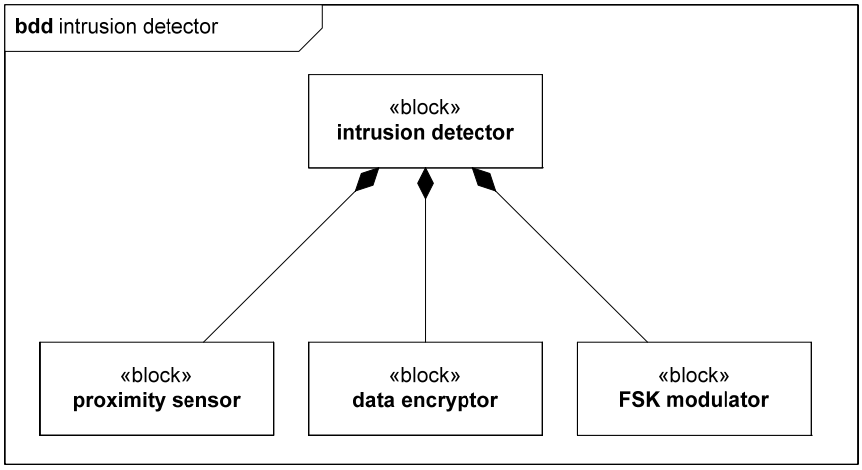

Figure 11 - Intrusion detector block definition diagram

6.2.2 SPD Power Node

The SPD Power Node is based on a Xilinx ML507 Evaluation Platform with a Virtex-5 FPGA.

The nodes that compose this Power Node are depicted in the figure that follows. It includes:

- a legacy FSK demodulator is a digital demodulator working with a clock of 32 kHz and demodulating 12 bits modulated data into 8 bits demodulated ones
- a SPD specific demodulator, providing the legacy demodulator SPD capabilities, such as metrics and discovery. The system has several metric values (dependability level, number of failure occurred, number of successful recovery occurred, etc.) and it answers over IP protocol on recognition request incoming from the network layer. It is able to provide upper layers with the class it belongs to, the subclass specific features, the kind of demodulation, the carrier, the sampling rate and other information useful to identify the node
- a dependability module, contains error detection and recovery. The system can recognize a fault condition (with a hardware based detection subsystem) and a plausibility evaluation subsystem. If a fault is recognized the system tries to restore the damaged feature reconfiguring such a part of the FPGA using the Partial Reconfiguration Feature
- data decryption: before modulating data, the system encrypts the data using a 64 bit fixed key and blowfish algorithm. This is a good compromise between robustness, liability and resources consumption
- reconfiguration, using partial FPGA reconfiguration for implementation of a new demodulation core, with a different carrier. The partial reconfiguration is also used to dynamically adjust the system. If the modulator switches for any reason from 2 kHz carrier to 1 kHz carrier, the system automatically recognizes the carrier has changed and adjust itself reconfiguring the part of FPGA given to demodulator with a new partial bitstream implementing the new required demodulator
- fault injector, triggered by a push button. A simple push button simulates a fault injection trigger. When an external agent presses the button the demodulator is reconfigured by partial

reconfiguration with a new copy of the bitstream, containing fault in its code, and the demodulator halts

- pSHIELD interface. In this case, it consists on a web server, providing a web page through HTTP, and XML information regarding the node identification, status, metrics, capabilities and function responses (distance to nearest object and alarms)

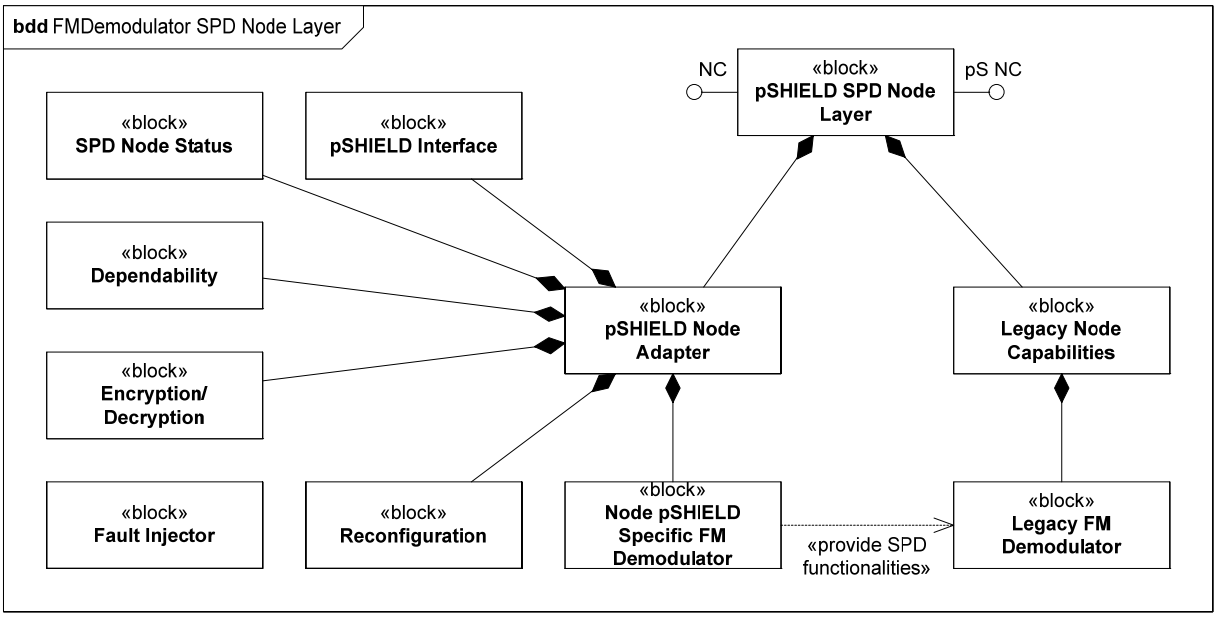

Figure 12 - FSK Demodulator SPD Node block definition diagram

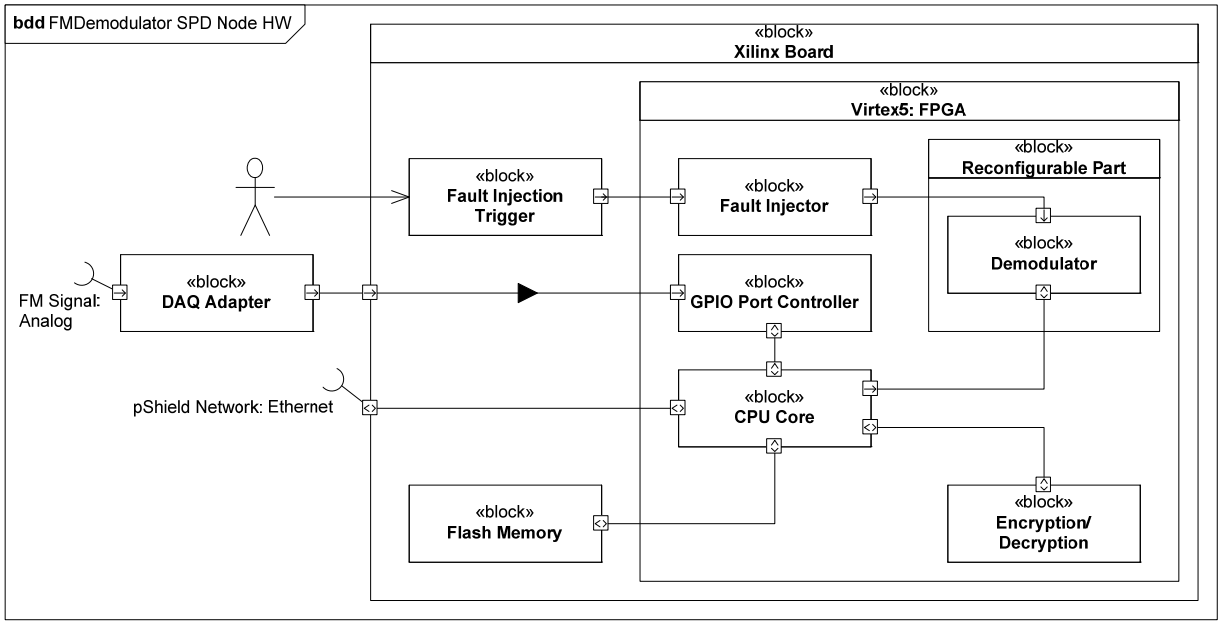

Figure 13 - FSK Demodulator SPD Node hardware modules

6.3 Interface with pSHIELD Network

The interface with pSHIELD middleware layer consists of requests to the Node services of capabilities, as presented in the figure below.

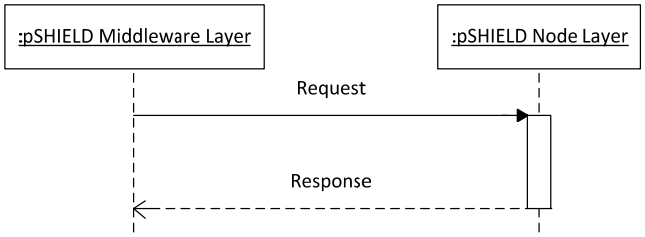

Figure 14 - Interface with middleware

However, for the sake of this demonstrator, this interface is implemented as an HTTP protocol (below figure). The Middleware is emulated by a control center, having the possibilities to receive information and sending control requests.

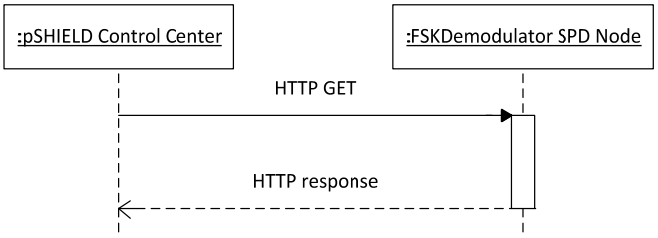

Figure 15 - Interface with middleware demonstrated with HTTP protocol and a Control Center

The FSK Demodulator SPD Power Node on request sends HTTP responses with updated information consisting of:

- Identification of the node, for system discovery
  - An ID and a Name
- Capabilities of the node, for system composition
  - Such as CPU model and frequency, RAM size, Error Detection mechanisms, Error Recovery mechanisms, Demodulation algorithm, Decryption algorithm, etc.
- Status of the node and all its components
  - Node status (running, starting, recovering, stopped, etc.), SPD level, status of different components, such as the decryption or the demodulation modules.
- Metrics information from all the components of the node
  - Errors detected, errors recovered, decrypted frames, decryption errors, demodulated frames, demodulation errors, reconfiguration requests, etc.
- Responses of the Node
  - Distance to the nearest object, intrusion alarm

This information is embedded in the web page in an XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<xml id="spdnode" style="display:none;">
 <identification>
  <id>001</id>
  <name>FSKDemodulator</name>
 </identification>
 <capabilities>
  <cpu>
   <model>PPC</model>
        ...
  <dependability>
   <errordetection>watchdogtimer</errordetection>
        ...
  <legacycapability>
    <type>FSKdemodulation</type>
  </legacycapability>
```

```
<commands>
    <command>
     <service>reset</service>
     <values>on,off</values>
          ...
 <status>
  <nodestatus>running</nodestatus>
  <SPDlevel>2</SPDlevel>
          ...
 <metrics>
  <dependability>
   <errorsdetected>2</errorsdetected>
          ...
 <responses>
  <distance>30</distance>
 </responses>
</xml>
```

The control center may send following requests:

- Turn decryption on or off
- Reset the node
- Reconfigure the FPGA to the bitstream with the demodulator with the other carrier
- Turn the alarm on or off, resetting the current alarm information

6.4 Encrypted Communications

In order to further enhance the secure communication aspect of the nodes, a new cryptographic key exchange protocol has been designed for the SPD enabled nodes. This protocol applies to both the Micro Nodes as well as the Power Node. Current implementation of the key exchange protocol supports the Sun Spot Nodes and it can be adapted to the other node implementations as well.

The key exchange protocol, namely “Control Randomness Protocol”, dictates an alternative key exchange mechanism similar to the well know hybrid key exchange protocol. Like the hybrid key exchange protocol, it consists of two distinct phases that incorporate different cryptographic technologies. On the first phase, a public key cryptography scheme is used in order to exchange the bundle of symmetric keys that will be used on the second phase. During the second phase, those keys are being used as input for a symmetric key cryptography scheme that handles the actual data exchange.

The concept of controlled randomness i.e., having multiple active keys at any given time moment, offers superior security characteristics compared to conventional protocols. The system designer can reuse well-known cryptographic blocks in a novel way to achieve increased security with minimal hassle.

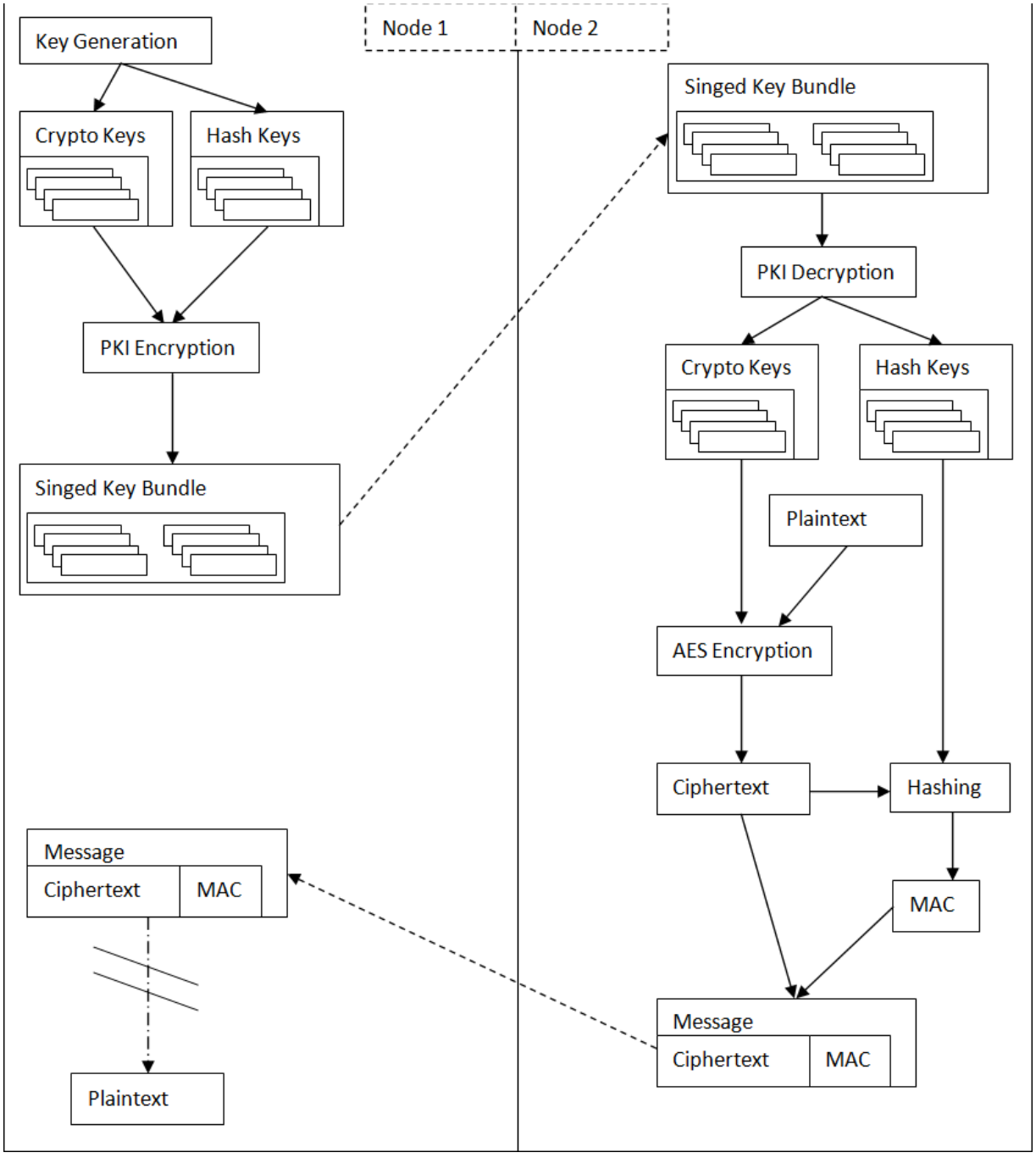

Figure 16 - the CRP encrypted communication protocol

In the respective figure, we can see the block diagram of the basic operations of the CRP protocol. The dashed lines denote the exchange of encrypted messages through the communication channel. The second part of the message exchange is the part where the actual payload is transferred and repeats for the lifetime of the cryptographic keys. The first part is where the cryptographic keys, and their respective hashing keys, are being exchanged. This part is computationally expensive and takes place only once per lifetime of the exchanged keys.

Like the well known hybrid key exchange scheme the actual cryptographic keys are being exchanged via the use of a public key cryptography scheme. In our case, instead of just one key, there are multiple keys that are being exchanged and are going to be used for the message encryption.

The CRP allows, in the above scenario, to extend the lifetime of each key way beyond the time of a conventional session. Further, it allows less frequent exchanges of messages in the control channel, since less keys are needed to achieve a specific security level for a specific timeframe

For the implementation of the CRP encryption protocol we used AES as the underlying symmetric encryption algorithm and SHA256 as the hashing algorithm. As described in D3.4, we experimented on various scenarios for different values for the number of cryptographic keys and the key exchange window and we see that, for certain combinations, the computational overhead is well under 5% while producing a significant increase to the lifetime of the active cryptographic keys during a session.

7 Cognitive Radio Network Prototype

7.1 Ambient Intelligence

The employment of sophisticated tools for data analysis in distributed or structurally complex systems requires the development of specific data fusion strategies to integrate the heterogeneous information coming from the environmental sensors. In such a framework, intelligence distribution is one of the most interesting research fields: the logical tasks are partitioned in real time between the various architecture components: intelligent sensors, intermediate nodes and remote control centers. Typical tasks such as context analysis and recognition are decomposed into hierarchical chains of subtasks. Each logical block of such functional chains receives as inputs the data produced by the lower block and produces a representation of the environment at a higher abstraction level. The latter will be supplied to the higher level blocks, and so on, obeying strict temporal constraints.

The data fusion process can therefore be sequentially assigned to different levels of the architecture in a distributed way, in order to output an overall representation of the environment and specific indications of situations of interest.

A typical description of a security system can be done in terms of a hierarchical tree structure, where sensors, elaboration nodes and remote control centers are connected through heterogeneous communication channels. Within such a structure each sensor contributes to the global monitoring by gathering specific data. Since sensors are presently provided with (narrow) elaboration skills, raw environmental data are locally analyzed and aggregated metadata are sent to the intermediate elaboration nodes.

Control centers are spots of the architecture where all relevant environmental data are conveyed by the intermediate elaboration nodes and gathered in real time in order to be usable (possibly by human operators by means of specific interfaces) in order to face out of the ordinary situations with targeted actions.

In this report the role of elaboration nodes in such architectures is analyzed. Advantages (in implementation and application) deriving from the use of biologically inspired cognitive models are pointed out. The application of ambient intelligence to pSHIELD is eventually depicted.

7.2 Intelligent Systems

Intelligent systems are defined as such (Velastin et al. 2004) whenever they are designed to integrate the Environmental Intelligence Paradigm (Remagnino 2005) with the traditional security applications. The

Paradigm defines as fundamental properties of “intelligent” systems the capacity of context analysis and intelligent distribution.

7.2.1 Context Analysis

Many recent works have been having as main objective the realization of tools for the automatic analysis of the context; such analysis is usually focused on the recognition of the behavior of the people in the scene, since the ability to classify behavioral information is of fundamental importance in managing security and in preventing critical situations in risky environments. Some examples of Ambient Intelligence applications to security systems are: classification of interpersonal and person-to-object interactions, threat recognition (Velastin et al. 2004) (Moncrieff 2008) (Brdiczka 2006).

The utilization of movement schemes for behavioral analysis and anomaly recognition is an effective approach, as it allows to accurately establishing the movement of various entities within the environment. Trajectories are grouped by means of specific grouping techniques; appropriate behavioral models are hence constructed.

A model for different human activities must be constructed taking into account the natural variability of human behavior. Each person performs the very same activities in a different way. Moreover, some actions can acquire different meanings depending on the overall global situation. Therefore, being not realistic to model human behavior deterministically, appropriate probabilistic models for the description of activities and interaction are often used.

7.2.2 Distributed Intelligence

Distributed intelligence, a distinguishing feature of intelligent systems, is a crucial factor for concurrent optimization of the communication channel between the blocks of the architecture and of the global elaboration skills of the systems. The possibility of data analysis at low levels in the architecture implies, for instance, less data load at higher levels and denies overloads or delays that could easily occur in case all the elaboration was concentrated in a single spot in the architecture. Such a solution also provides more robustness by means of delocalization and redundancy of the elaboration activity: distributed systems are, as a matter of fact, less susceptible to single components breakdowns.

Typical tasks can be decomposed in a chain of logical modules, organized in a hierarchical structure: low level modules produce as outputs the meta-data needed by the higher level modules. This way, starting from raw data (non-processed data), a representation of the environment at a higher abstraction level is obtained at each level of the architecture. Such decomposition, originally proposed in (Marcenaro et al. 2001) defines the intelligence distribution paradigm in terms of logical modules allocation within the different physical elements of the architecture, with autonomous data elaboration abilities.

The modularity of the functionalities of intelligent systems and their allocation in different subsystems must however guarantee a quality in the analysis, which must at least be equal to the case where the elaboration is located entirely in one only architecture block. It is therefore necessary for the modules to communicate to each other, independent of their physical location and the link between them (e.g. wireless or wired). Moreover, modules distribution and the necessity of saving the data generated from them, makes it necessary to memorize representations of detected events in suitable structure bounds to the physical device.

There are three typologies of modules, defined at different abstraction levels: representation modules (information’s elaboration tasks: the output is a higher level symbolic representation of the data than the input), recognition modules (algorithms compare input data with a set of models) and communication modules (which produce a codified representation of the input data, suitable for their transmission).

Such modules are the logical components by which collect together the different functionalities, namely the parameterization alphabet of the applicative middleware for the security. This allows the architecture

to work in a dynamic, reliable and flexible way. The chains of modules can be loaded by means of the functions for the dynamical management of the network resources, in the control centers or when activating the functionalities requested by the user, or whenever changes in the state of the network occur.

7.3 Cognitive Systems

Two limitations of the intermediate elaboration modules in an intelligent system are their passivity and the inability of learning based on experience.

In fact, despite the development of specific applications capable of semantic context analysis, such systems are passive, since they are not designed to work over the environment to solve threat situations. Usually, the chain of modules of context analysis located within the elaboration nodes, produces, in case of anomalies, specific alarms which are sent to human operators for decision making and action.

Cognitive systems can overcome these limitations by means of a cognitive cycle (sensing-analysis-decision-action). Cognitive systems indeed implement a model which imitates the brain functionalities and not only are able to correctly analyze the meaning different situation, but can also to act consequently after a decision. A cognitive system has the capability of interacting in a closed cycle with the outside world by means of the actuators present in the environment.

The cognitive system has an internal model which describes the actuators related to itself and the action they can make towards the environment (embodied cognition).

Cognitive systems make use of a learning phase to codify within appropriate data structures the behavioral models, based on experience. To be precise, the information stored is the one concerning the relations between changes in the state of the system and changes in the outside world (and vice-versa). This way, a cognitive system can recognize some situations and forecast, through an inference mechanism, their future development without any information on rules.

A cognitive system can also learn from experience the decisional models of a human operator, based on his actions as a reaction to specific environmental situations. The knowledge acquired is used to model specific automatic decision routines based on context meta-data coming from the chain of logical blocks of analysis. One can therefore define automatic decision blocks at different abstraction levels based on the information concerning the state of the system, the current events, the predicted events and the classification of the current scenario.

A cognitive system than overcomes the typical limitations of simple intelligent systems by adding to the architecture of the system appropriate logical blocks devoted to decision and learning.

7.4 The cognitive model

Intelligent systems are defined as such (Velastin et al. 2004) whenever they are designed to integrate the Environmental Intelligence Paradigm (Remagnino 2005) with the traditional security applications. The Paradigm defines as fundamental properties of “intelligent” systems the capacity of context analysis and intelligent distribution.

Cognitive systems are based on a neurophysiological model of reasoning and awareness (Damasio 2000). In this model, a cognitive entity is described as a complex system which is able to learn incrementally – on the basis of experience – relations between themselves and the external world. Neuroscientific conceptualization of cerebral human functions defines two specific devices, called proto-self and proto-core, which are devoted to the monitoring and management of the internal state of the entity and of the external world respectively. The possibility of gaining access to its own internal state

(self-consciousness) is for the cognitive entity as necessary as the ability of analyzing the environment. According to this model the sensors available to a cognitive entity can be divided into endo-sensors (or proto sensors) and eso-sensor (or core sensors) depending on whether they are used for internal or external states monitoring.

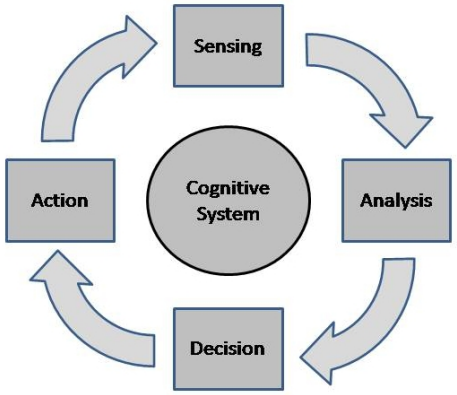

Figure 17 - Cognitive Cycle

The behavior of a cognitive entity interacting with the world is described by the cognitive cycle and can be divided (below figure) in four fundamental steps. Sensing, Analysis, Decision, Action. These steps represent, as time flows, an infinite sequence, since the state (internal and external) which is perceived at each step is (directly or indirectly) influenced by past Actions made by the cognitive entity itself.

Therefore, the conceptual architecture of a cognitive entity is made of four logical blocks:

Sensing: a cognitive system constantly gets information about the core- and self-states by means of endo- ad eso-sensors.

Analysis: the data coming from the sensors are fused in order to obtain a common description of the external world as well as the internal state of the cognitive system. Input data are than analyzed to detect events, which can in turn be either proto events (εP), relative to significant changes in the internal state of the system or core (εC), relative to changes in the external world. From such data, a cognitive entity is able to create a model of probability distributions of proto and core events, p(ϵ_t^P |ϵ_(t-1)^P) and p(ϵ_t^C |ϵ_(t-1)^C). This model (first order neural pattern) does not account for possible interactions between core and proto events and can be regarded as a couple of Dynamic Bayesian Networks (proto-DBN and core-DBN).

Decision: according to the experience of the cognitive system (obtained through a codification of past events filtered through appropriate data structures) and to the analysis of the current internal and external states X_P and X_C, the system selects the most appropriate strategy ST in order to get the desired configuration of the system {X_P,X_C}. The target configurations {X_P,X_C} are selected in order to get stability (homeostasis) with respect to specific behavioral models (learned or available).

Action: this module implements the active interaction of the system towards the surrounding environment: an appropriate action a_i is selected based on the strategy ST chosen during the previous step. Such an action is executed on the environment or on the system itself by means of suitable specific actuators.

7.5 The pSHIELD Simulator

The pSHIELD simulator has been developed in the cognitive framework described above.

7.5.1 Scenario

The scenario consist in a number of entities (agents) carrying a mobile device which is able to transmit and receive data at 3 different frequencies (namely 900, 1800 and 1900 MHz) to a centralized control center. The agents move randomly throughout a radio-disturbed environment, where randomly placed jammers emit a disturbing signal. The jammers can be either fixed or moving and their emitted signal follows the Rayleigh distribution with fixed parameters. Fixed jammers positions and characteristics are stored in an XML file, which is loaded in the setup stage together with the map of the ground.

The mobile devices periodically send a single radio data to the control center, where a running cognitive node receives and elaborates it. Also, a periodical polling is performed by the agents to question the node, which answers back.

A radio data sent by an agent contains the following pieces of information:

- *Position of the agent (x,y) on the mapped ground*: this is generated by a trajectories simulator. It simulates a GPS sensor on the mobile device. If a video monitoring of the ground area is available, positioning data coming from a tracker can be possibly fused to GPS data to obtain a better position estimation.
- *Frequency of transmission*: this can be chosen among the three available frequencies at the beginning of the simulation.
- *Power of the transmitted signal*: fixed.
- *Power of the signal received from the node*: this depends on the distance and it is calculated through FSPL. Also, it can be disturbed by jammers.
- *Possibly detected jammers' estimated power*: each jammer has a typical radius (coded in the XML configuration file) of influence, inside which the agent can measure its power.
- *ID of possible neighbor agents* (within a fixed sensing radius).

A slightly different scenario can be also set by introducing a moving jammer: an agent carrying a jamming device can be introduced in the scene. Such an intruder-agent differs from the others as he obviously disturbs communications to the node. Also, he communicates a false GPS survey to the node.

7.5.2 Cognitive model application

The radio data reception represents, from the node point of view, the sensing logical block of the cognitive cycle. The agents' mobile terminals are the sensors which monitor the environment sending a radio survey (radio sensors) and a positioning piece of information (GPS sensor).

The node then analyzes all the data received from each agent, both singularly and collectively. For each agent, the signal-to-noise and distortion ratio (SINAD) of the received data packet is computed. Also, the relative positions the agents are compared, on the basis of the datum sent by an agent himself and of the fused data sent by the agents in the sensing range. By means of a voting algorithm, rankings are assigned to the IDs of each agent. The intruder's position and ID are worked out as soon as enough information is gathered, based on such rankings.

In the decision stage, the SINAD datum is compared to an acceptable (fixed to 10 dB) threshold. If the communication with an agent turns out to be too disturbed, a suitable strategy ST is chosen to schedule a change in frequency transmission.

The action block provides a change in the state of the system. As already explained, this module implements the active interaction of the system towards the surrounding environment or towards itself: the action of changing frequency is selected based on the strategy ST chosen during the previous step.

Such an action is executed on the system itself by means of suitable actuators, namely the agents. Actually, as already pointed out, through a periodical polling, the agents themselves ask the node for information: however this does not change the heart of the matter.

The detection of the intruder does not trigger a decision and a subsequent action in the cognitive cycle. The information relative to the false agent is simply communicated to an interface. Such an interface could simply be in a control center, or could display data on the mobile devices, thus leaving the decision step under human control. Alternatively, a strategy could be implemented to be learned by the cognitive node in a future perspective.

8 Semantic Model Prototype

The Semantic Model Prototype belongs in the Middleware demonstrators, coupled with the Common Criteria reasoner. While the Core SPD services provide the basic functionalities of the pSHIELD Middleware, the Semantic Model provides the information necessary to take decisions and drive them.

The Semantic Model (OWL file) that has been developed for demonstration purposes is structured in this way:

- A section to represent system's components
- A section to represent *functional properties*
- A section to represent *SPD relevant information:* attributes, threats, means of mitigation
- Attributes to identify relations between system and functionalities
- Attributes to quantify SPD level
- A reasoner to perform the SPD composition according to the Common Criteria rules defined in WP2

On an implementation perspective, the following classes have been developed:

- For the structural ontology: System, Element, Hardware, SPD Component
- For the functional ontology: SPDFunctionality, GeneralFunctionality, Connector, SPDCompositionSpecification
- For the attribute ontology: SPDConcept, SPDAttribute, SPDThreat, SPDMean

And the reasoner (semantic engine) has been structured in the following way:

- At design time (offline) the semantic engine helps along the configuration of a system architecture, by discovering proper combinations of SPD modules, according to the corresponding semantic model of modules and composability rules picked out from an offline repository (catalogue); at run time (online), changes in the state of the system trigger the semantic engine to devise new compositions, based on knowledge of modules that at the moment are active in the system (possibly discovered at run time), in order to guarantee the prearranged overall SPD level. (Synthesis)
- At run time (online), the semantic engine oversees the current value of the overall SPD level as the state of the system evolves in time (Analysis)

Regarding the Common Criteria based operations that have been identified in the proposal for the aggregation of SPD metrics and to the requirements of ontological SPD modelling, a number of suitable mixes of rules and ontology actions has been used to develop the aggregation features, including, but not limited to: MIN, OR and MEAN operations, Redundancy configuration.

The semantic model derived so far has been properly instantiated in the final integrated demonstrator. The prototype delivered for WP5 is in the form of OWL file, i.e. an xml file:

Figure 18 - pSHIELD OWL (XML File)

9 Middleware Prototype for the demonstration of composability

9.1 The pSHIELD Simulator

The essential role of middleware is to manage the complexity and heterogeneity of distributed infrastructures. On the one hand, middleware offers programming abstractions that hide some of the complexities of building a distributed application. On the other hand, there is a complex software infrastructure that implements these abstractions. With very few exceptions, this infrastructure tends to have a large footprint.

pSHIELD middleware has a modular structure to achieve interoperability between heterogeneous parts to guarantee the desired SPD functionality level. The effective and efficient realization of such modular, interoperable, large-scale software components is facilitated by SOA (service oriented architecture) because it provides a standardized architecture for modular systems, for creating new functionality from existing building blocks, and for enabling communication between heterogeneous component models represented in the abstract by services.

pSHIELD's core SPD services are a set of mandatory basic SPD functionalities provided by a pSHIELD Middleware Adapter in terms of pSHIELD enabling middleware services. The core SPD services aim to provide a SPD middleware environment to actuate the decisions taken by the pSHIELD Overlay and to monitor the Node, Network and Middleware SPD functionalities of the Embedded System Devices under the pSHIELD Middleware Adapter control. The following core SPD services are provided:

- service discovery
- service composition
- service orchestration

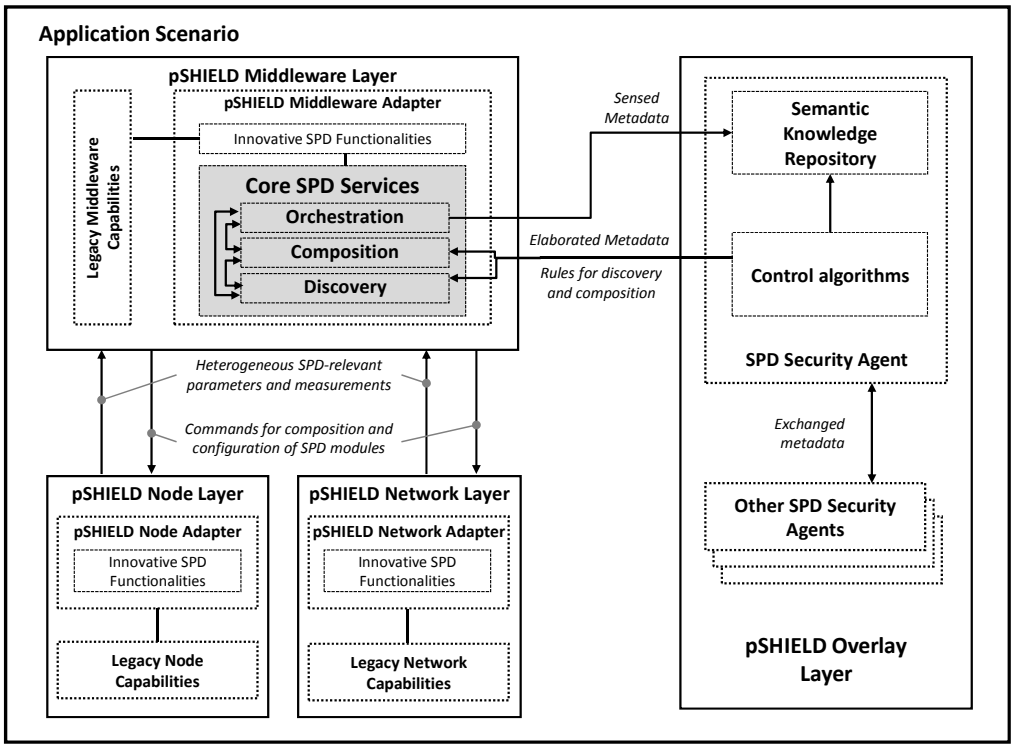

Figure 19 - Core SPD services in the pSHIELD functional component architecture

Service discovery allows any pSHIELD Middleware Adapter to discover the available SPD functionalities and services over heterogeneous environment, networks and technologies that are achievable by the pSHIELD Embedded System Device where it is running. Indeed the pSHIELD secure service discovery uses a variety of discovery protocols (such as SLP, SSDP, NDP, DNS, SDP, UDDI) to harvest over the interconnected Embedded System Devices (ESDs) all the available SPD services, functionalities, resources and information that can be composed to improve the SPD level of the whole system. In order to properly work, a discovery process must tackle also a secure and dependable service registration, service description and service filtering. The service registration consists in advertising in a secure and trusted manner the available SPD services. The advertisement of each service is represented by its formal description and it is known in literature as service description. The registered services are discovered whenever their description matches with the query associated to the discovery process, the matching process is also known in literature as service filtering. On the light of the above a SPD services discovery framework is needed as a core SPD functionality of a pSHIELD Middleware Adapter. Once the available SPD services have been discovered, they must be prepared to be executed, assuring that the dependencies and all the services preconditions are validated. In order to manage this phase, a service composition process is needed.

Service composition is in charge to select those atomic SPD services that, once composed, provide a complex and integrated SPD functionality that is essential to guarantee the required SPD level. The service composition is a pSHIELD Middleware Adapter functionality that cooperates with the pSHIELD Overlay in order to apply the configuration strategy decided by the Control Algorithms residing in the

pSHIELD Security Agent. While the Overlay works on a technology independent fashion composing the best configuration of aggregated SPD functionalities, the service composition takes into account more technology dependent SPD functionalities at Node, Network and Middleware layers. If the Overlay decides that a specific SPD configuration of the SPD services must executed, on the basis of the services' description, capabilities and requirements, the service composition process ensures that all the dependencies, configuration and pre-conditions associated to that service are validated in order to make all the atomic SPD services to work properly once composed.

Service orchestration is in charge to deploy, execute and continuously monitor those SPD services which have been discovered and composed. This is part of the pSHIELD Middleware Adapter functionality. While service composition works "off-line" triggered by an event or by the pSHIELD Overlay, service orchestration works "on-line" and is continuously operating in background to monitor the SPD status of the running services.

The Orchestration, Composition and Discovery functionalities are the enablers of the decisions taken by the pSHIELD Security Agent Control Algorithms residing in the pSHIELD Overlay. The mutual interoperation between the pSHIELD Middleware Adapter and the pSHIELD Security Agent enables the pSHIELD Composability concept.

9.2 Middleware prototype

The demonstration of Composability of SPD components is based on the implementation of the pSHIELD Middleware using the OSGI framework.

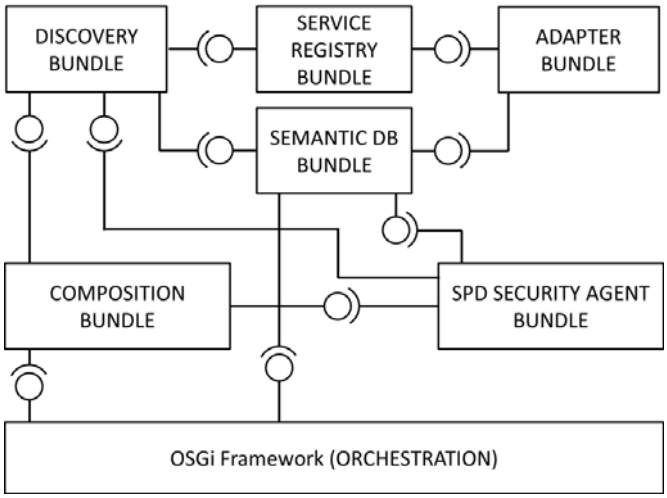

Figure 20 - High level Core SPD Services prototype architecture

The prototype architecture derives directly from the architecture described in D5.2 "SPD middleware and overlay functionalities prototype". Each pSHIELD component is mapped into an OSGi bundle and, when needed, decoupled into a composition of interoperating bundles each providing a specific functionality.

Discovery Bundle: The discovery bundle structure is depicted in the following figure:

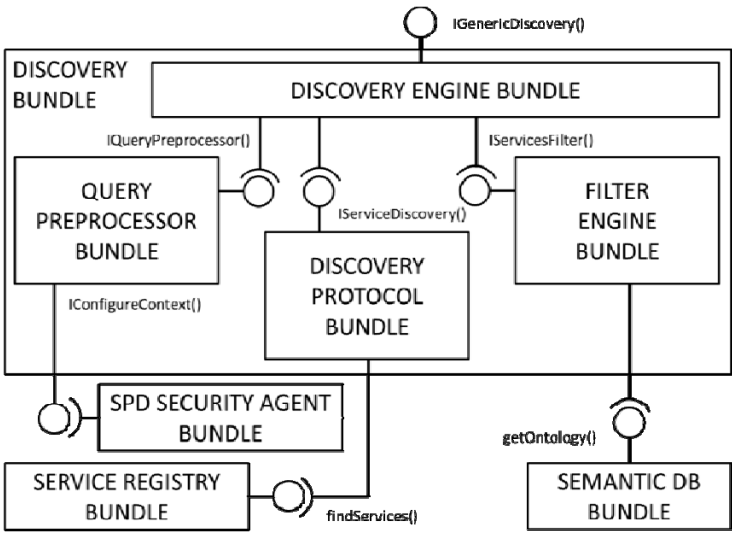

Figure 21 - Discovery Bundle structure

As explained in the previous sections, the Discovery Bundle is composed by the following bundles:

- **Discovery Engine Bundle**: it is in charge to handle the queries coming from the *IGenericDiscovery()* interface. The Discovery Engine Bundle manages the whole discovery process and activates the different functionalities of the Discovery service. It calls the *IQueryPreprocessor()* interface to enrich semantically and contextually the query. After that the query is sent to the different underlying discovery protocols, by means of the *IServiceDiscovery()* interface, to harvest over the interconnected systems all the available SPD components. Finally the list of discovered services is sent to the Filter Engine Bundle using the *IServicesFilter()* interface to discard those components not matching with the enriched query.
- **Query Preprocessor Bundle:** it is in charge to enrich the query sent by the Discovery Engine with semantic information related to the peculiar context. The query pre-processor can be configured by the SPD Security Agent to take care of the current environmental situation using the *IConfigureContext()* interface;
- **Discovery Protocol Bundle:** it is in charge to securely discover all the available SPD components description stored in the Service Registry Bundle, using a the *findServices()* interface;
- **Filter Engine Bundle:** it is in charge to semantically match the query with the descriptions of the discovered SPD components. In order to perform the semantic filtering, the Filter Engine can retrieve from the Semantic DB the information associated to the SPD components, by means of the *getOntology()* interface.

Service Registry Bundle: The Service Registry Bundle structure is depicted in the following figure:

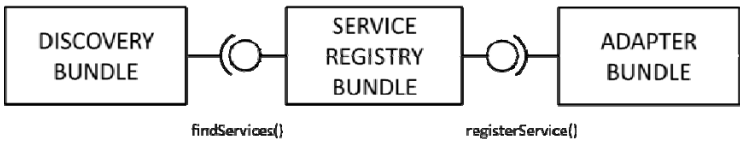

Figure 22 – Service Registry Bundle

- **Service Registry Bundle:** it is in charge to store the bundle (i.e. SPD component) description in terms of provided functionalities, interfaces, semantic references, etc.. Any pSHIELD Node, Network or Middleware layer component can be registered here to be discovered by its own proper pSHIELD Adapter. The Adapter registers each bundle as a service, using the *registerService()* interface. The Service Registry provides the services entries information to the Discovery Bundle by means of the *findServices()* interface.

Adapter Bundle: The Adapter Bundle structure is depicted in the following figure:

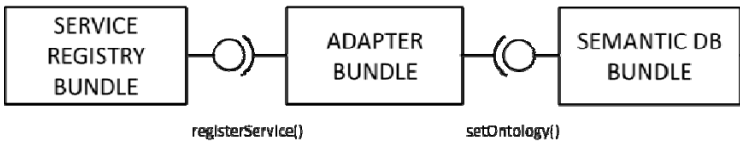

Figure 23 – Adapter Bundle

- **Adapter Bundle**: it represents a generic (Node, Network or Middleware) pSHIELD Adapter for any type of legacy SPD functionality. The Adapter Bundle is in charge to:
  1. Provide an Innovative SPD functionality interacting with the underlying legacy services, capabilities and resources;
  2. register the provided Innovative SPD Functionality in the Service Registry using the *registerService()* interface;
  3. publish the semantic description of the Innovative SPD Functionality in the Semantic DB using the *setOntology()* interface;

Semantic DB Bundle: The Semantic DB Bundle structure is depicted in the following figure:

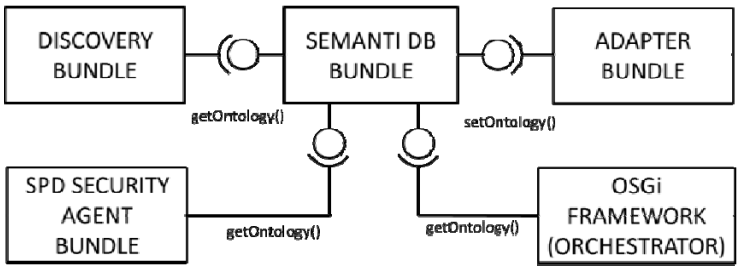

Figure 24 - Semantic DB Bundle

- **Semantic DB Bundle**: it is in charge to store properly the semantic set by each Adapter Bundle through the *setOntology()* interface. The stored ontologies contains all the information to compose the available Innovative SPD functionalities. The Semantic DB Bundle provide access to the ontologies through the *getOntology()* interface.

Composition Bundle: The Composition Bundle structure is depicted in the following figure:

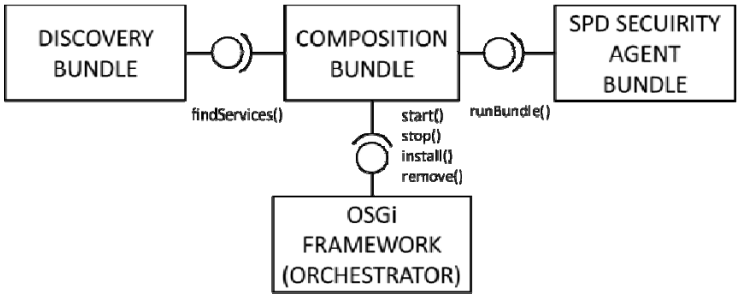

Figure 25 – Composition Bundle

- **Composition Bundle**: it is in charge to compose the discovered bundles accordingly with the composition rules determined by the SPD Security Agent. Once the SPD Security Agent communicates through the *runBundle()* interface the necessity to run a composed functionality, the Composition Bundle use the *findServices()* interface to discover any suitable SPD component to be composed. Then the Composition Bundle compose the available bundles (taking care of the inter-bundle dependencies and the API-IMPL relationships) and uses the *start()*, *stop()*, *install()* and *remove()* interfaces provides by the Orchestrator (that is the OSGi framework itself).

SPD Security Agent Bundle: The SPD Security Agent Bundle structure is depicted in the following figure:

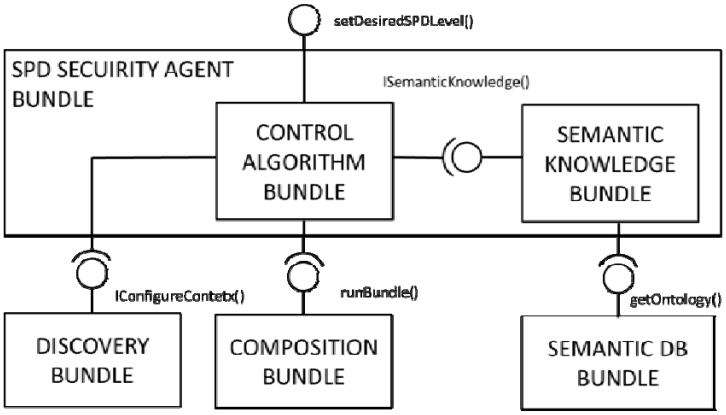

Figure 26 - SPD Security Agent Bundle

As explained in the previous sections, the SPD Security Agent is composed by the following bundles:

- **Semantic Knowledge Bundle:** it is in charge to get the semantic description of the available services using the *getOntology()* interface and to make inference on their semantic model to extract the SPD level of their composition;
- **Control Algorithm Bundle**: it is in charge to evaluate the best control strategy for the whole system in terms of proper configuration rules both for the Discovery and the Composition Bundle, respectively through the *IConfigureContext()* and *runBundle()* interfaces. The Control Algorithm can influence which services can be discovered configuring the query preprocessor and can influence the composition process limiting the composition only to the best SPD functionalities that can assure the desired SPD level.

The prototype infrastructure has been deployed into a real OSGi framework. A screenshot of the OSGi control panel is reported below:

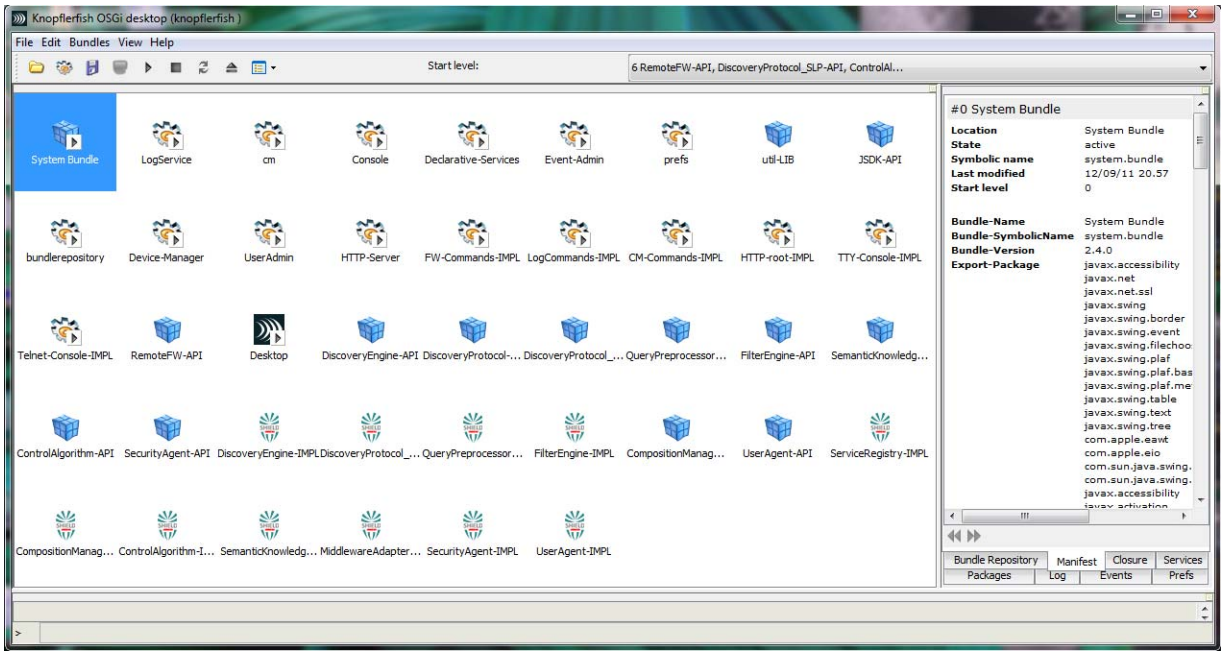

Figure 27 - OSGi framework

In this screenshot all the above introduced bundles are shown correctly running in a OSGi environment. The Core SPD Services prototype will be used to setup the pSHIELD pilot Demonstrator by adding proper pSHIELD Adapters to communicate with meaningful components of the Railway Application Scenario.

9.3 Composability Concept Demonstration

The middleware prototype for the demonstration of composability is composed by a central unit connected by means of a ciphered wireless network to remote sensors. These components are supplied with the related configuration manuals. In this platform the assets to protect are data sent by remote sensors to central unit, where data is recorded inside the central unit itself.

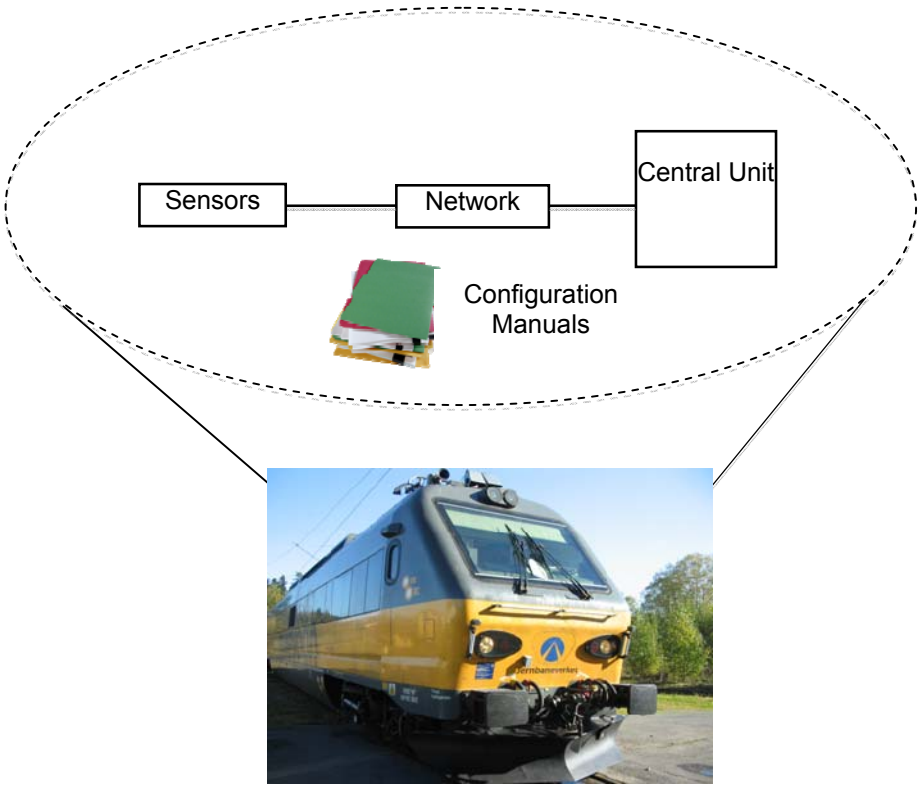

Figure 28 – Platform Structure

This platform is further detailed in section 4 of Deliverable 6.2 “Platform validation and verification”.

The main elements are the Central Unit and the Wireless Sensor Network, where the main elements are implemented. The joint operation of the pSHIELD overlay, the pSHIELD Core SPD services (discovery, composition and orchestration) and the pSHIELD middleware layer (that interact with the network and the node layer too) apply as a closed loop system, where the Current SPD level measured by the pSHIELD middleware is continuously compared with the Desired SPD Level by the Overlay. The Overlay applies for configuration rule to react against any potential SPD gap between the desired level and the current level. The configuration rules are then enforced into the system of embedded systems by the Core SPD services and applied concretely by the pSHIELD adapters at middleware, network and node layer.

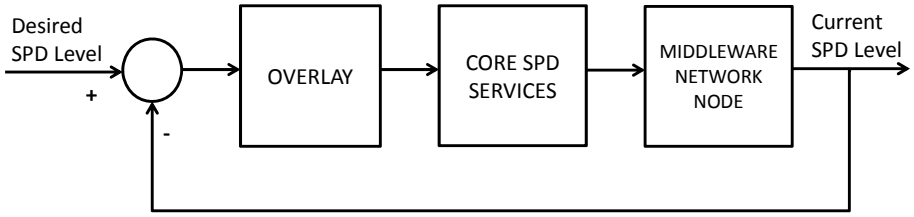

Figure 29 - Closed-loop SPD level control

The results of this approach, as well as the description of how the integration between the middleware software and the nodes was accomplished is detailed in Deliverable 6.2 “Platform validation and verification”.

10 Policy-based management and Hybrid-Automata model

10.1 Policy Based approach

A typical PBM architecture includes two types of nodes at the button node layer, categorized based on their capabilities in terms of processing power and capacity, i.e., power nodes and simple sensor nodes. Power nodes are described to be more resourceful while sensor nodes are typically seen as resource constrained devices. Upper supporting layers constitute Network, Middleware and Application layers while agents in a vertical overlay monitor/tune those layers.

Given the aforementioned architecture, a PDPs and PEPs from a typical PBM architecture can be mapped naturally to power and sensor nodes respectively (following figure).

On the lower layer, sensor nodes being the managed resources are considered as policy enforcers, i.e., PEPs. The latter, based on the XACML model, should enforce authorization decisions and handle affiliated obligations specified by applicable rules. PEPs can support local policy storage in order to comply with COPS-PR mode of operation hence the provision of a local PIP although not compulsory. However, this depends on the capabilities of deployed sensor nodes whether they can afford a form of local policy storage and decision making. Moreover, power nodes are those nodes that are more resourceful than the sensor nodes which make them natural decision making points able to process/translate policies and deduce rules to be enforced by affiliated PEPs. The COPS protocol can govern the communication between PDPs and affiliated PEPs but not exclusively, as SNMP is an option as well (where an LPIP is no more required).

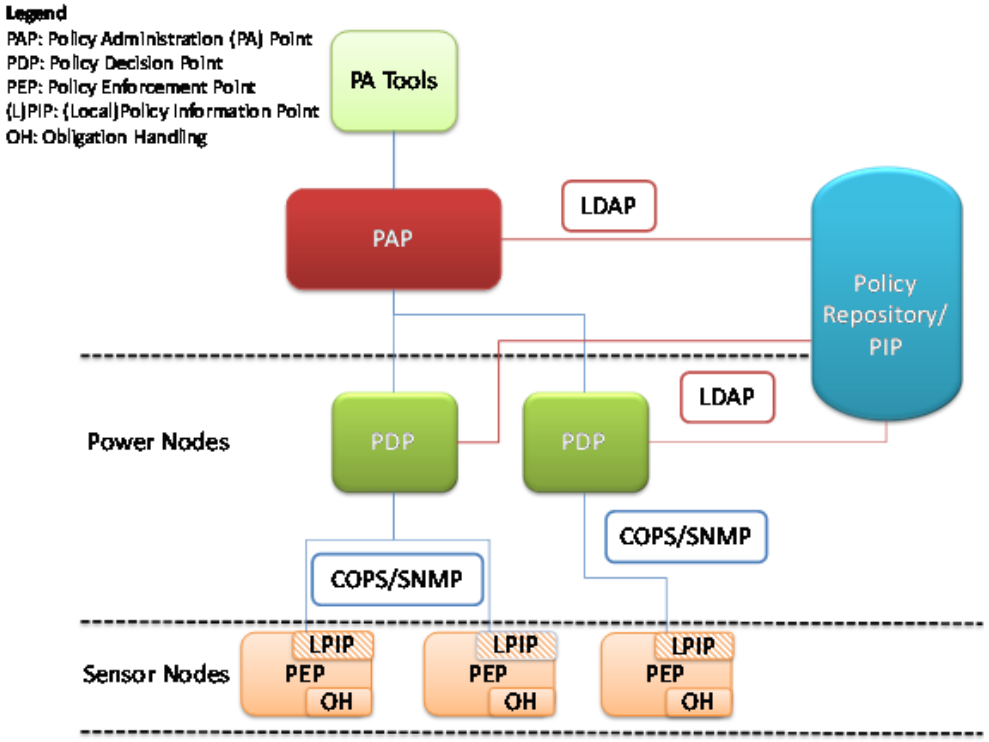

Figure 30 - PBM Mapping

A group of PDPs can access the repository of policies, (i.e., PIP) in order to retrieve needed polices for evaluation. This is done through LDAP that is a protocol suited for lightweight read-intensive operations allowing for directory access from different platforms and locations. The policy repository is managed solely by the policy administrator point (PAP). Also, PAP is responsible for providing policy authorizing tools besides management and control capabilities. These could include creation, termination, activation, listing, amending and synchronizing policies.

Concerning pSHIELD's main scenario where a monitoring and access control system is put in place to oversee rail-transported hazardous materials, the above PBM is considered suitable. Locking and access control mechanism in addition to installed sensors can be seen as PEPs where the central control unit in the train carriage can be seen as a PDP with local access to PIP. Moreover, the central command centre overseeing the operation of the monitoring system is seen as a PAP with policy administration tools and repository support. The PIP is expected to be distributed which allows a given PDP to access it locally where a PAP can manage such a distributed PIP through LDAP.

This analysis will not be integrated in the final demonstrator due to the limited resources dedicated to this activity and the adaptation effort needed to integrate a policy based management in the OSGI Framework (integration is possible on a technological perspective, but requires time and resources: for that reason it will be one of the objective of the nSHIELD project).

10.2 Hybrid-Automata approach

This approach concerns the formalization, by means of Hybrid Automata Theory, of some control laws that are supposed to drive the SPD composition.

This concept is simple but effective: the Common Criteria approach defines a standard methodology to compose elements with precise quantification of their SPD level. Since the solution of the composition problem is not always unique, we can enrich this composition by setting further rules that allows discriminating from one configuration to the other. This can be done by creating a dynamic model of the system and verifying, with respect to pre-defined objective functions, the most convenient configuration.

Two different approaches have been demonstrated to validate this theory, both supported by numerical simulations (that constitute the final output).

10.2.1 Prototype a – Static Approach with Simple Optimization

The first, simple, approach, is based on the following steps:

At first the system "state" is identified, i.e. the set of active components (node, protocols or applications). A state is a screenshot of the system in a specific condition (for example with the node E switched on) and with the dynamics associated to this condition (for example the evolution of the node's power consumption).

The selected dynamics considered for this model constitutes the so-called context information: since the SPD is controlled via the common criteria approach, we need to insert into the model variables that could be significant to control (optimize) the evolution of the system. They could be, for example, the power consumption, the computational resources utilization, the bandwidth utilization, and so on.

The state identified in this step is depicted in the following figure:

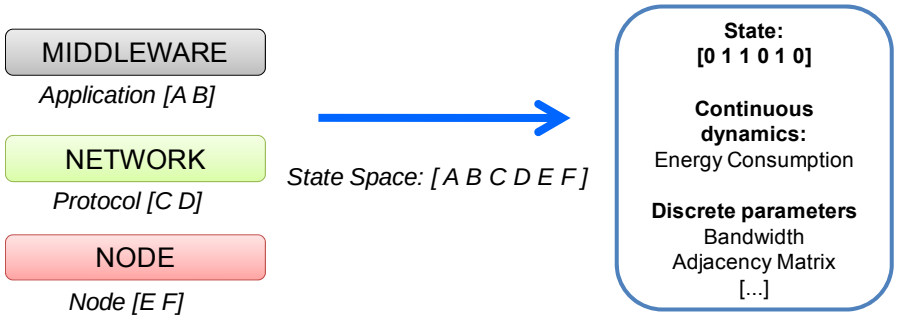

Figure 31 – Single State representation

Secondly, different states are concatenated to obtain the universe of all the possible condition of the system: this is an enumeration of configurations. For example in a system with two nodes, two network protocols and two middleware services with 8 states (at least one component must be active).

Q = {[101010], [101001], [100110], [100101], [011010], [011001],[010110], [010101]}.

The result is depicted in the following figure:

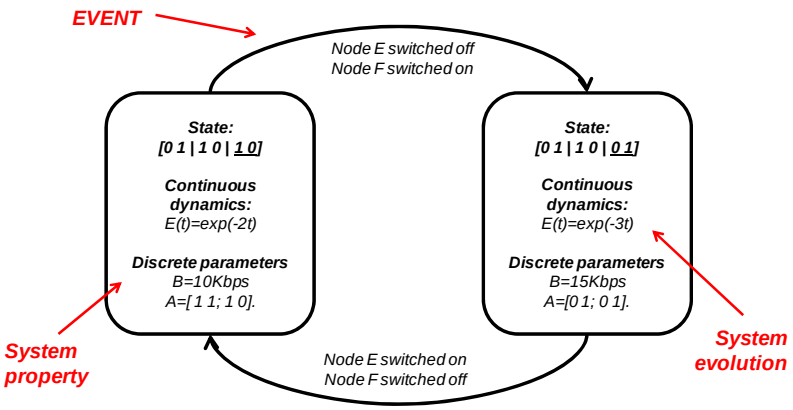

Figure 32 - Hybrid Automata to describe all the possible configurations

The transition can be voluntary and expected (control action) or not (due to fault) but in any case each event is captured and in every moment it is possible to check the status (and evolution) of the system:

D = {switch configuration$_1$, fault$_1$, …, switch configuration$_n$, fault$_n$}.

The third step is the identification of the internal variables (and dynamics) to control. For the pilot project a simple case is considered where:

- the relevant dynamic is the power consumption of the system in a specific configuration and
- the amount of bandwidth provided by the network layer.

These variables have opposite behaviours (higher bandwidth, higher power consumption) so the purpose of the control algorithm is to choose the configuration that optimizes one of them.

This scenario has been implemented in Matlab-Simulink (see figure below) and is composed by two nodes with two different dynamics for the power consumption and for bandwidth utilization. It is important to notice that both these configurations should be valid SPD configurations (see CC approach).

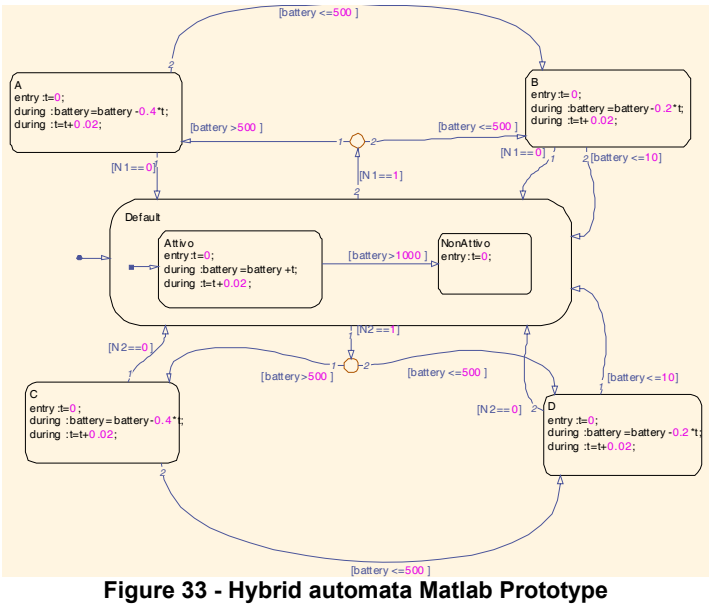

Figure 33 - Hybrid automata Matlab Prototype

10.2.2 Prototype b – Operating conditions approach with MPC Control

The second prototype aims at being more efficient and flexible to cope with the scalability issues that in a complex system may arise. This has been obtained by clustering the representation of the configurations in a more restricted environment: the operating conditions. Given an Embedded System (pSHIELD Node) it is possible to identify a set with elements (battery, buffers, CPU) that can be associated to an operating conditions: a buffer can be saturated, full or empty; a CPU can be idle, working or overloaded; a battery can be full or empty. All these components can also be broken. The combination and aggregation of these conditions allows creating an exhaustive model of a pSHIELD node, as depicted in the figure below. The aggregation is possible, since some behaviours of the components have the same effect of the system (if the CPU or the Buffer is full, the result is always the impossibility of processing data).

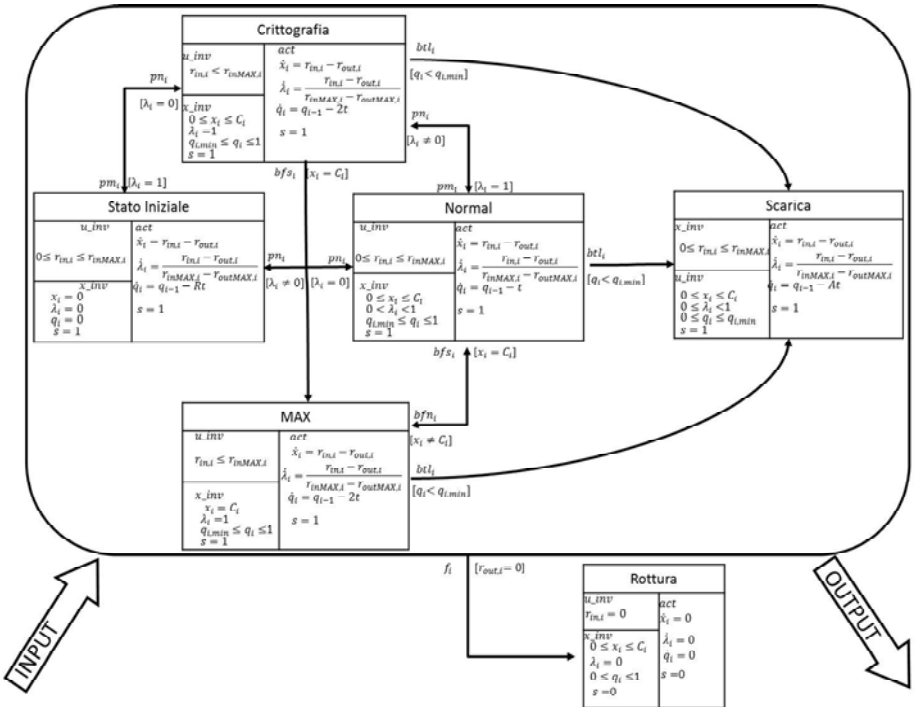

Figure 34 - Hybrid Automata representing the pSHIELD node

At this point the problem of scalability of composition is solved, since the introduction of a new node in the system doesn't imply an exponential increase in the model size, but a linear growth (6 states for each additional node and 4 states for each additional network layer).

Last, but not least, interesting control algorithms can be applied to the system model due to its formulation by means of these operating conditions (see for example the work of Bemporad [8] and [9]). In particular for the pSHIELD purposes the framework developed in [9], based on Model Predictive Control (MPC), has been considered to verify the effectiveness of the Hybrid Automata approach.

For the simulations it has been used the Matlab Toolbox for Hybrid System with the default configuration (standard MPC problem). The Objective of the control algorithm has been to maximize the amount of data processed by the node while preserving the battery and leaving a certain amount of "reserved" resources for potential emergency tasks.

The Hybrid Automata prototypes will not be integrated in the final demonstrator because their role is mainly to validate the control law and not to “implement” the control law in the OSGI environment. However further studies in nSHIELD project will lead to the translation of the Matlab simulations file into a C++ or Java language to perform the same task directly in some software routines at middleware level.

As for the Policy based Management the “potential” integration is assured by the software abstraction and definition of proper interfaces (even if it will be carried out in nSHIELD).

11 Security integration across heterogeneous platforms

The architecture used here is based on sensor-as-a-service approach [44]. The architecture is composed of three layers: (i) the Real-world access layer, (ii) the Semantic Overlay layer, and (iii) the Service Virtualization layer. It follows the SOA concept to address all the capabilities needed to respond to the dynamics of a real-time IoT infrastructure. SOA exercises a classic request/response communication pattern, and relationship between service and its consumer is synchronous due to the nature of request/response. This only classic demand-based passive approach is not suitable for the interoperable rail information system use case because events play a vital role in the use case under implementation. Making sense of IoT events and performing a course of action in response to these events is the highly demanding capability of any IoT service framework. Here, we will provide the detail of each layer of IoT virtualization framework.

The real-world access layer provides an interface with underlying IoT cloud. It implies an adapter oriented approach to address the technical diversity regarding sensor types and communication mechanisms. One of the main goals of this layer is to get real-world information and carry it to the upper layer for further processing. It receives the sensor events and dispatches them to an event manager by using a callback message pattern, where messages are sent asynchronously to the receivers that later process the messages and take appropriate action. This layer can also transfer action messages from upper layer and then select appropriate adapters to deliver it towards smart objects (i.e., Sun SPOT node). The node periodically broadcast its capabilities. The implementation uses the comma-separated list to exchange the node capabilities information. The implementation purposefully avoids the use of XML for capabilities to reduce the overhead that comes with XML as it is too much verbose.

The semantic overlay provides the semantic model of underlying devices by maintaining IoT ontology, the sensor ontology, an event ontology, and the service access policies. It facilitates CRUD (create, read, update and delete) operation on knowledge base. The layer also supports both persistent and in-memory storage. The in-memory caching mechanism keeps the last observation of smart objects in order to boost the performance of the framework.

The goal of the service virtualization layer is to expose the functional aspects of smart objects of reference use case in the form of services. The layer aims at delivering requester the information they look for based on their access rights. The layer performs various tasks such as notifying all the subscribers of a specific sensor event.

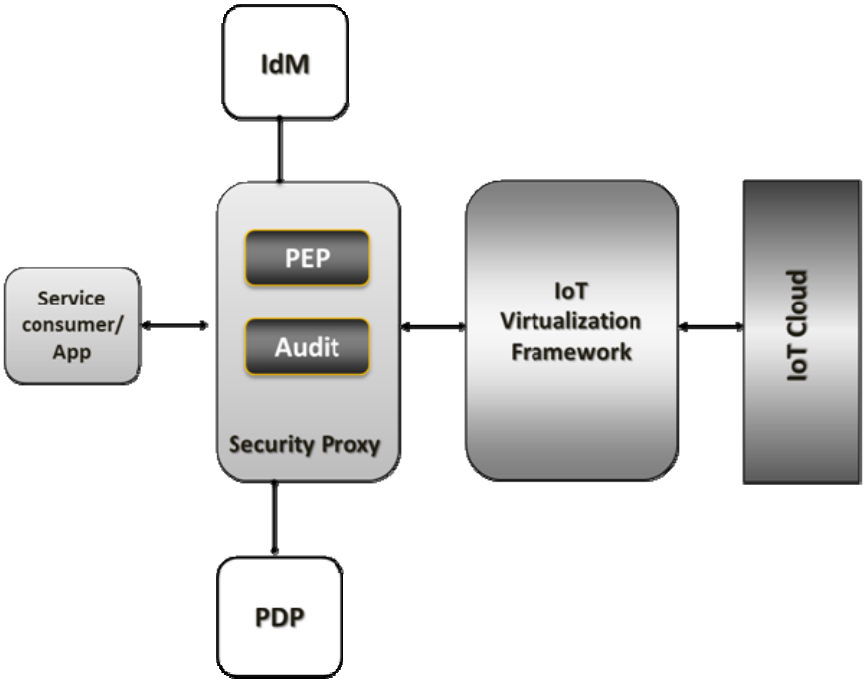

Figure 35 – Sensor-as-a-service architecture

The implementation could support both the pull and push based interaction. The following is a sequence diagram of push based service interaction.

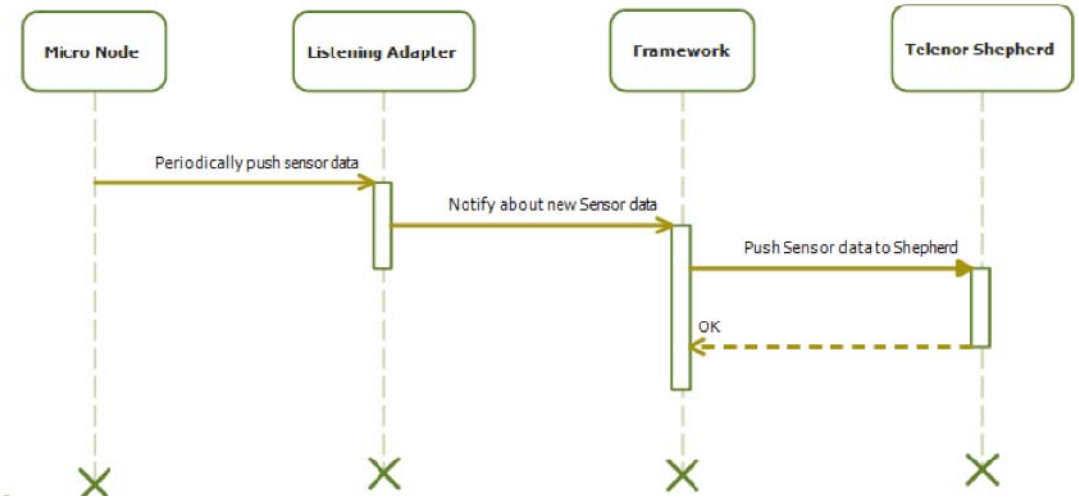

Figure 36 - Push based service interaction

In order to satisfy seemingly divergent security requirements, we exploit the service oriented security proxy and externalize IoT security operations. The security proxy consists of a policy enforcement point, an audit and a policy decision point. The proxy follows the principal of reverse-proxy but we tailor it to the

special characteristics of the reference use case. This will allow security proxy as a special purpose intermediary service, abstracting the security related technical details (i.e., policy enforcement) from service consumers and allowing distributed provisioning of common security services.

Heterogeneity of security caused by capabilities of different nodes, involved in the reference use case, is addressed by announcing node capabilities forehand. The implementation assumes that the Nano nodes do not support any security algorithm/protocols. However, each micro/personal node can support different security algorithms/protocol. Each micro node broadcasts its supported security algorithm/protocol in its periodical capabilities announcement message. The message is received by the framework that maintained this in ontology. Whenever, the communication adapter needs to communicate with a specific node it knows about the node security capabilities and uses the same security algorithm/protocols while communicating with a specific node. However, this approach requires the framework to support multiple number of security algorithms/protocols in order to be functional.

The security proxy runs as a web service on Apache Tomcat servlet container. It is developed using Apache Axis Web Service framework. The service call-out is logged through log4j. We formulated polices related with services using web ontology language (OWL) and semantic web rule language (SWRL). The prototype uses simple xml file for maintaining the user-defined polices. We use Pellet API to exploit reasoning capabilities for evaluating the access policies and determining the access decisions by executing SWRL rules.

12 Platform for heterogeneous Wireless Sensor networks

12.1 Technologies description: Wireless Sensor Network

The main purpose of a Wireless Sensor Network is to serve as an interface to the real world, providing physical information such as temperature, light, radiation, and others, to a computer system. WSN are expected to be a breakthrough in the way natural phenomena are observed: the accuracy of observations will be considerably improved, leading to a better understanding of the monitored environment. These networks have a simple structure: there are dozens up to 100s of elements, called ”sensor nodes” able to sense physical features of their surroundings or to monitoring a set of items. WSN nodes exchange information on environment in order to build a global view of the monitored items/regions which is made accessible to the external user through one or more gateway node, named base station or sink node [33]. Sensor nodes are often referred as smart sensors or smart dust because of their processing, power, and memory capabilities [34]. A WSN typically operates by stepping through the following phases:

- sensor nodes acquire sensed data
- data is locally processed
- data is routed in a multi-hop fashion
- data is delivered to the sink node
- data is forwarded by the sink node to a conventional network, e.g. Internet

The main drawback of WSN sensor nodes is the restricted resource of energy leading to limited lifetimes. This fact motivates attention and effort the research community has devoted to the development of low power consumption techniques, not only at MAC layer, but also at network and application layers.

WSN Requirements

Lifetime

In most application scenarios, a majority, if not the totality of the nodes are self-powered, and hence, in the best, they are able to survive for a limited time. The most common adopted lifetime metric is related to the time till a certain percentage of surviving nodes in the network falls below a given threshold.

Area Coverage

Area coverage is defined as the ratio between the number of up, running, and connected nodes at a given instant of time, over the number of initially deployed sensors. Due to the aging and wear out process of nodes, the area coverage is a decreasing function of the time. WSN applications define the minimal level of area coverage to assure so that the observed phenomenon can be monitored with acceptable confidence.

Timeliness

In environment monitoring applications is often required to correlate samples coming across different nodes in order to gather combined measurements. In this case, nodes must be synchronized in order to take part to the distributed computation correctly, i.e. by providing samples acquired within a bounded interval of time. The main mission of a typical WSN is to collect environmental data and to send.

Challenges

Although the technology for WSNs is relatively mature, and WSN have been employed in several pilot research applications, real large scale applications are completely lacking. This is in part due to a number of still unsolved problems afflicting WSNs.

Main challenges related to WSN implementation are reported in the following.

Energy Conservation

Because of the reduced size of the sensor nodes, the battery has low capacity and the available energy is very limited. Despite the scarcity of energy, the network is expected to operate for a relatively long time. Given that replacing/refilling batteries is usually impossible or very expensive, one of the primary challenges is to maximize the WSN lifetime while preserving acceptable performances. Low-quality communication WSNs are often deployed in harsh environments, and sometimes they operate under extreme weather conditions. In these situations, the quality of the radio communication might be extremely poor and performing the requested collective sensing task might become very difficult.

Operation in hostile environments

In many scenarios, WSN are expected to operate under critical environmental conditions, which translates in an accelerated failure rate of sensor nodes. Thus, it is essential that sensor nodes are carefully designed, and the WSN assessed under real failure assumptions. Furthermore, the protocols for network operation should be resilient to sensor fault, which must be considered in these scenarios a norm rather than an exception.

Security Attacks

As networks grow the vulnerability of network nodes to physical and software attack increases. Attackers can also obtain their own commodity sensor nodes and induce the network to accept them as legitimate

nodes, or they can claim multiple identities for an altered node. Once in control of a few nodes inside the network, the adversary can then mount a variety of attacks, for instance, falsification of sensor data, extraction of private sensed information from sensor network readings, and denial of service attacks. Therefore, routing protocols must be resilient against compromised nodes that behave maliciously. Ensuring that sensed information stays within the sensor network and is accessible only to trusted parties is an essential step toward achieving security. Data encryption and access control is one approach. Another is to restrict the network is ability to gather data at a detail level that could compromise privacy. In this case security comes at the price of a reduced lifetime due to the extra overhead induced in the network.

Maintenance Cost

The initial deployment and configuration is only the first step in the WSN lifecycle. In WSN where deployment is expected to surpass the lifetime of batteries, the total cost of management for a system may have more to do with the maintenance cost than the initial deployment cost. Throughout the lifetime of a deployment, nodes may be relocated or replaced due to outages, and discharged batteries. In addition, reintegrating the failed nodes adds further labor expenses. An approach to limit interventions would be to increase the lifetime by adopting a trigger-based sampling strategy: sensors start to acquire data only when given conditions are met. However, this approach introduces a further coordination problem among sensors, e.g. ,nodes monitoring the same area must agree on the triggered event, synchronize their clock, and start to sample data coordinately. Since access costs are dominant over in-situ costs, it is important, therefore i) to identify sources of maintenance related costs and to reduce them, and ii) to schedule maintenance so that once the network is accessed, a convenient number of nodes are maintained and hence, the overall maintenance cost optimized.

Lack of easy to commercialize applications

Nowadays, several chip makers and electronic companies started the production of sensor nodes. However, it is much more difficult for these companies to commercialize applications based on WSN. Selling applications, instead of relatively cheap sensors, would be much more profitable for industry. Unfortunately, most sensor network application scenarios are very specific, and companies would have little or no profit in developing very specific applications, since the potential buyers would be very few.

In the next sections there will be the description the architecture of SENSIM-SEC platform.

12.2 SeNsIM-SEC platform architecture

In many cases a monitoring infrastructure should provide the integration of critical data with other types of information retrieved by the surrounding environment, with different security requirements, too. The architecture proposed is built upon SeNsIM (Sensor Networks Integration and Management) [31,35], a framework that was designed for integration of heterogeneous sensor networks based on the wrapper-mediator paradigm [32]. It provides a unified interface by which users can easily execute queries on the system to retrieve network information and elaborate sensor data. In SeNsIM each different network of the system is managed by a dedicated wrapper that is able to communicate with the specific underlying technology and acts as a connector for the mediator component; the mediator is responsible to properly format user requests and forward them to the different wrappers, this translates the incoming queries and injects them into the underlying networks, retrieves the results and passes them back to the mediator.

The communication between the mediator and the wrappers is carried out by means of XML files, written according to a standard format and containing information about the structure of the underlying networks, the user-defined query parameters and the retrieved results.

In order to make the integration effective, each wrapper explores and monitors the local sensor networks (discovery phase) and sends to the mediator an appropriate description of the related information according to a common data model, in a struct.xml file. The mediator organizes such information and keeps a unique view of all systems in order to satisfy user or application queries.

By means of the mediator GUI, a user can specify a query by indicating the desired values and other relevant parameters such as sample period and query duration; the mediator translates user requests and builds a query, sending it to the wrappers responsible of the target network/s.

The retrieved results are collected and written in a result.xml file by each involved wrapper, and sent back to the mediator, which encodes and stores sensed data into its database in order to make them available for elaboration or data fusion. In order to reduce the TCP communication overhead, a retrieval interval can be specified by the end user, identifying the time interval by which the wrappers may collect query results and send them back to the mediator (in a single result.xml file), thus not having to wait until the query duration elapses. The retrieval interval can be tuned in order to control the trade off between system performances and the acceptable delay in obtaining the results from wrappers.

The figure below shows the interactions that take place during normal operation between such components according to two main usage scenarios, registration and querying:

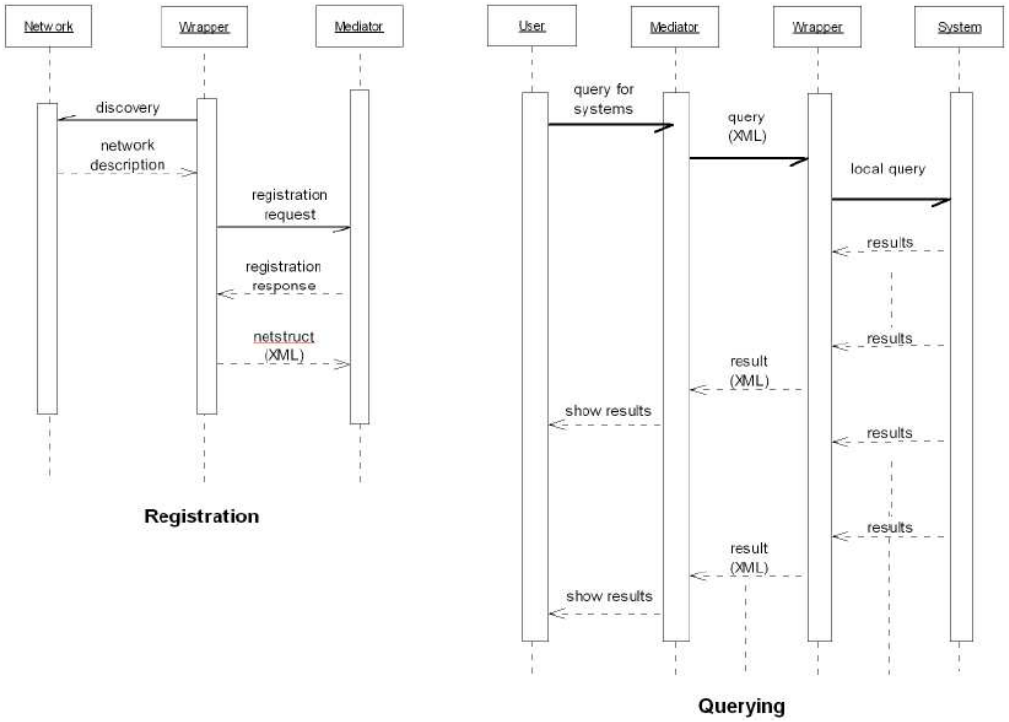

Figure 37 - Registration and querying

For the *Registration,a*ny wrapper needs to register itself before communicating with a mediator. At first, each wrapper creates an XML document describing the system as a whole; this is done by analyzing the sensor system (i.e. by injecting a discovery query), and generating the appropriate XML to represent the system. Then a wrapper sends a registration request message to the mediator, that verifies the possibility of including a new system in the framework and sends a response message to the wrapper. If the

registration request is accepted, the wrapper sends the XML document to the mediator, which stores the related information in a DB.

The *querying* process starts when a user sends a query request through the mediator user interface after having selected the destination of the query (a network or a specific sensor, among those connected to the framework trough the wrappers). The mediator takes the query parameters and creates an XML document which sends to the appropriate wrapper. The wrapper extracts the parameters by parsing the XML document and executes the query on the local system. The query results are grouped by the wrapper, which also creates another XML document, and periodically send it to the mediator. Finally the mediator extracts the results from the XML and shows them to the user.

Each query message (sent to a given wrapper) is characterized by the following information:

- a *query identifier* that allows to univocally distinguish the query inside the system
- the network/sensor system identifier
- a *type* that identifies the kind of a query (monitoring query or, when possible, event query)
- some *temporal parameters* such as sample time, duration and results retrieval interval
- one or more *destination* in terms of single sensors, cluster of sensors, group of sensors or the whole sensor system
- the set of *predicates or sensing functions* (i.e. light, temperature, acceleration, etc...) that have to be invoked on the specified data source

The SeNsIM architecture was extended in order to design a heterogeneous sensor network infrastructure able to manage the heterogeneity not only in the technology aspects but also in the different security requirements (see figure below).

It's considered one of the most adopted OS for sensors, TinyOS [36], installed on each simple motes (sensor nodes) of the network and on the sync node (master node) that acts as a connector towards the application level, forwarding the incoming queries to the motes and returning results back to the querying wrapper. The security requirements of the sensor network have been achieved by implementing a hybrid cryptosystem based on the ECC [35,37] primitives for key agreement operations and digital signature generation and verification. In order to realize such operations it's been implemented two different applications to be run respectively on the master and the motes nodes, based on the operating system components and on the WM-ECC library primitives. Further details on the cryptosystem and the flexible wrapper component are given in the Deliverable 6.2.

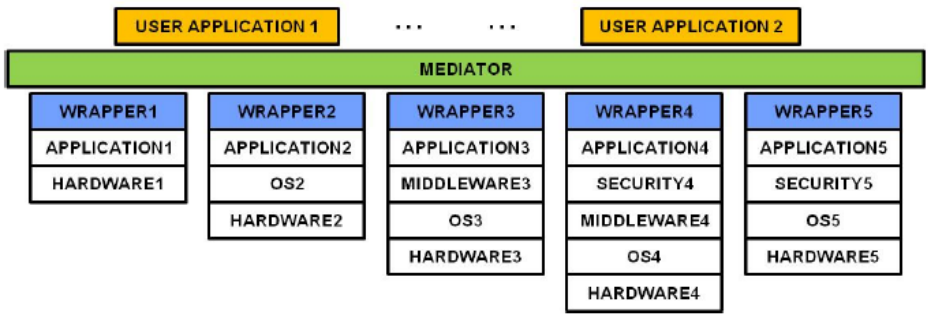

Figure 38 - Sensim-Sec Architecture

13 Synergies across heterogeneous modules

Though pSHIELD had the goal of creating prototypical pilot implementations, all sub-developments focused on interoperability. This interoperability covers various aspects, ranging from sensor descriptions to applications on the sensor data:

a) sensor interfaces
b) semantic description of sensor data (yet to come)
c) security description/metrics (test - yet to harmonise)
d) semantic middleware
e) OSGI-based service discovery, supporting composability
f) prototypical demonstration of interoperability on service/application layer
g) standardization

Sensor interfaces
This unit has defined centre interfaces group after 90, Micro, personal, and pollen nodes. Our assumption is that nano nodes will have their own specific format which normally can't be changed. Thus integration of nano nodes has always to be accompanied by an overlay, which we suggest is based on a semantic description of the sensor data.

An example of such an overlay is provided through the Telenor objects platform Shepherd, where either an HTTP or a Java-based interface is used to include nano sensors.

Semantic descriptions of sensor data
Sensor ML is aiming at providing a semantic description of sensors, allowing for specification of the output format and other sensor characteristics [SensorML].

Lightweight semantics is seen as being in the solution to bring semantic description down to sensor level. One of the most popular examples of lightweight semantics are micro-formats that use existing XHTML techniques [Khare]. RDFa is another notable lightweight semantic technology that allow semantic markup to be included within XHTML [Adida]. RDFa is more powerful than micro-formats as it can include expressive ontologies. Ostermaier et al. proposed a sensor micro formats using HTML syntax [Ostermaier]. Apart from these suggestions we are not aware of advanced semantic standards for sensors.

Security description/metrices (test - yet to harmonise)
In pSHIELD we are aiming at the security descriptions through metrics, addressing both the SPD metrics of the system and the threats. The main focus has been to demonstrate that the principle of metrics will work and that metrics can accurately described both the system and the threats. The project has established the metrics, but standardised way of measuring and describing the metrics is subject to the follow-on project nSHIELD. Thus transferring metrics from sensor level to application level is yet to be shown.

Semantic middleware
Semantic technologies are seen as the integrator for heterogeneous systems. In pSHIELD we have adopted this suggestion and developed middleware, which supports the integration of heterogeneous components. With respect to this middleware, we have developed a set of application interfaces (APIs),

allowing us to connect sensor data to platforms, policies, and applications. This part is well documented in the WP5 deliverables.

OSGI-based service discovery, supporting composability
One of the goals of pSHIELD was to show the service composability. Service composability is based on the availability and registry of services in the system. We decided to use the OSGI framework, as OSGI has been a straightforward way of addressing service discovery. All power nodes developed in the project support OSGI bundles. OSGI bundles also exist for Android Smartphones, which in our case are typical representatives of personal nodes.

Prototypical demonstration of interoperability on service/application layer
Our goal in the prototypical demonstration of interoperable security was to demonstrate that security features are forwarded across platforms. Through the integration with the Telenor Objects Shepherd™ platform we could demonstrate portability of some security features like sensor-ID and dependability. These sensor data are available for the policy engines making use of the standardized interfaces to Shepherd™.

The follow-on activity nSHIELD will address how policy decisions will affect the configuration of the sensor subsystem.

Standardization
In this section we have already introduced some standards related to sensor semantics, i.e. SensorML, RDFa and Microformats. We also pointed out that the Telenor Objects Shepherd™ platform is an instance of the ETSI TS 102.690 standard. Thus pSHIELD has demonstrated the use of existing standards where possible. However, we have also recognized that standardized approaches for "semantically described security" are recognized by the market, but is far from becoming a standard. Thus we assume that further work has to be done to create an ecosystem of companies being interested in the secure-interoperability, before we can bring the outcomes to standardization.

14 Conclusions

D6.1 represents consortium's effort to present the components that will comprise the pSHIELD demonstration platform and will evaluate the basic concepts of the project. Apart from the apparent goal of developing innovative technologies that will stand as added values in the framework of Embedded Systems with high SPD values, WP6 aims also to open the way to the integration of interoperable sub-systems with the aforementioned characteristics. And although at this stage a fully unified platform from high integrated components is difficult to be visualized, also due to the pre-mature stage of the newly introduced components, potential abstractions of interoperability exist, as the previous chapter attempted to highlight. During the development of local modules and individual components, the consortium changes its focus towards more tangible results, preferring to consolidate each technical step, rather than present an holistic solution. We hope to exploit this knowledge in further expanding the idea of SHIELD, facing the technical challenges and opportunities for related research and development that are ahead.

15 References

[1] A. J. Goldsmith and S. B. Wicker, "Design Challenges for Energy-Constrained Ad Hoc Wireless Networks," *IEEE Wireless Communications Magazine*, pp. 8–27, Aug. 2002.
[2] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-Layer Design for Wireless Networks," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 74–80, Oct. 2003
[3] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. New Jersey: Prentice Hall, 1992.

[4] V. T. Raisinghani and S. Iyer, “Cross-Layer Design Optimizations in Wireless Protocol Stacks,” *Computer Communications*, vol. 27, pp. 720–724, 2004.
[5] A. S. Tanenbaum, *Computer Networks*, 3rd ed. Prentice-Hall, Inc., 1996.
[6] L. Larzon, U. Bodin, and O. Schelen, “Hints and Notifications,” in *Proc. IEEE Wireless Communications and Networking Conference (WCNC’02)*, Orlando, 2002.
[7] G. Xylomenos and G. C. Polyzos, “Quality of service support over multiservice wireless internet links,” *Computer Networks*, vol. 37, no. 5, pp. 601–615, 2001.
[8] Q. Wang and M. A. Abu-Rgheff, “Cross-Layer Signalling for Next-Generation Wireless Systems,” in *Proc. IEEE Wireless Communications and Networking Conference (WCNC’03)*, New Orleans, 2003.
[9] M. Conti, G. Maselli, G. Turi, and S. Giordano, “Cross-Layering in Mobile Ad Hoc Network Design,” *IEEE Computer Magazine*, pp. 48–51, Feb. 2004.
[10] Soon-Hyeok Choi, Dewayne E. Perry and Scott M. Nettles: “A Software Architecture for Cross-LayerWireless Network Adaptations”, The University of Texas at Austin Austin, Texas
[11] Vijay T. Raisinghani, Sridhar Iyer: “Cross Layer Feedback Architecture for Mobile Device Protocol Stacks”,
[12] Vineet Srivastava and Mehul Motani Srivastava: “Cross-Layer Design: A Survey and the Road Ahead”, IEEE Communications Magazine, Dec 2005.
[13] Giovanni Giambene and Sastri Kota: “Cross-layer protocol optimization for satellite communications networks: A survey”, Int. J. Satell. Commun. Network. 2006
[14] Qi Wang and Mosa Mi Abu-Rgheff: “A Multi-Layer Mobility Management Architecture Using Cross-Layer Signalling Interactions” http://www.cis.udel.edu/~yackoski/cross/qwang_epmcc03_paper.pdf
[15] R. Winter et al., “CrossTalk: A Data Dissemination-Based Crosslayer Architecture or Mobile Ad Hoc Networks,”
[16] V. T. Raisinghani and S. Iyer: “ECLAIR: An efficient cross layer architecture for wireless protocol stacks”, WWC2004,
[17] Yana Bi, Mei Song, Junde Song: “Seamless mobility Using Mobile IPv6”, Publication Year: 2005
[18] Shantidev Mohanty and Ian F. Akyildiz: “A Cross-Layer (Layer 2 + 3) Handoff Management Protocol for Next-Generation Wireless Systems” IEEE Transactions on Mobile Computing, vol. 5, no. 10, Oct 2006
[19] Melhus, I.,Gayraud, T., Nivor, F., Gineste, M., Arnal, F., Pietrabissa, A., Linghang Fan: “SATSIX Cross-layer Architecture” Publication Year: 2008 , Page(s): 203 – 207
[20] Srivastava, V., Motani, M.: “The Road Ahead for Cross-Layer Design”, Publication Year: 2005 , Page(s): 551 – 560.
[21] IETF: Policy Framework [Online], http://datatracker.ietf.org/wg/policy/charter/
[22] Verma D.C.: “Simplifying network administration using policy-based management”, IEEE Network, 2002, pp. 20-26
[23] ETSI DES 282 001 V0.0.1 (2006-09)
[24] M. Al-Kuwaiti et al., “A Comparative Analysis of Network Dependability, Fault-tolerance, Reliability, Security, and Survivability”, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 11, NO. 2, SECOND QUARTER 2009
[25] T. Charles Clancy, Nathan Goergen, “Security in Cognitive Radio Networks: Threats and Mitigation”
[26] S.C.Lingareddy et al., “Wireless Information Security Based on Cognitive Approaches”, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.12, pp. 49-54, December 2009
[27] Jan Peleska, “Formal Methods and the Development of Dependable Systems”\
[28] Martin ARNDT, “Towards a pan European architecture for cooperative systems”, ETSI Status on Standardization, 2009
[29] AbdelNasir Alshamsi, Takamichi Saito, “A Technical Comparison of IPSec and SSL”, Tokyo University of Technology
[30] Your Electronics Open Source, “Embedded Systems in SDR and Cognitive Radio”, http://dev.emcelettronica.com/

[31] V. Casola, A. Gaglione, and A. Mazzeo, A reference architecture for sensor networks integration and management, 2009. In IEEE Proceedings of GSN09, Oxford, July 2009
[32] G. Wiederhold, Mediators in the Architecture of Future Information Systems, In IEEE Computer XXV(3), pp. 38-49, 1992
[33] Kirk Martinez, Jane K. Hart, and Royan Ong. Environmental sensor networks. Computer, 37(8):50–56, 2004
[34] B. Warneke, M. Last, B. Liebowitz, and K.S.J. Pister. Smart dust: communicating with a cubic-millimeter computer. Computer, 34(1):44–51, Jan 2001
[35] V. Casola, A. De Benedictis, A. Mazzeo and N. Mazzocca, SeNsIM-SEC: security in heterogeneous sensor networks, May 2011, SARSSI2011
[36] TinyOS Project, URL: http://www.tinyos.net
[37] H.Wang, B. Sheng, C.C. Tan and Qun Li, WM-ECC: an Elliptic Curve Cryptograph Suite on Sensor Motes, Technical report, Oct. 30, 2007
[38] Damasio A. (2000), “The feeling of what happens - body, emotion and the rise of consciousness”, Harvest Books
[39] Brdiczka O., P.C. Chen, S. Zaidenberg, P. Reignier and J.L. Crowley (2006), “Automatic Acquisition of Context Models and its Application to Video Surveillance”, International Conference in Pattern Recognition, 1175-1178, DOI:10.1109/ICPR.2006.292
[40] Marcenaro L., Oberti F., Foresti G., Regazzoni C. (2001), “Distributed architectures and logical-task decomposition in multimedia surveillance systems”, Proceedings of the IEEE (10) (October 2001) 1419-1440
[41] Moncrieff S., S. Venkatesh e G. West (2008), “Context aware privacy in visual surveillance”, International Conference in Pattern Recognition, 1-4, DOI:10.1109/ICPR.2008.4761616
[42] Remagnino P. e G.L. FORESTI (2005) Ambient Intelligence: A New Multidisciplinary Paradigm, Machine Vision and Applications, IEEE Transactions on Systems, Man and Cybernetics - Part A, 35, 1-6, DOI:10.1109/TSMCA.2004.838456
[43] Velastin S., L. Khoudour, B.P.L. Lo, J. Sun and M.A. Vicensio-Silve (2004) Prismatica: A multi-sensor surveillance system for public transport network, 12th IEE Road Transport Information and Control Conference, 19-25
[44] Sarfraz Alam, Josef Noll, “Enabling Sensor as Virtual Services through Lightweight Sensor Description”, in the proceeding of fourth International Conference on Sensor Technologies and Applications (SENSORCOMM 2010), July 18 - 25, 2010 - Venice/Mestre, Italy, pp. 564-569, ISBN: 978-0-7695-4096-2
[45] Sensor model language (SensorML). (2005)[online], http://www.opengeospatial.org/standards/sensorml
[46] B. Adida, M. Birbeck (2008). RDFa primer: Bridging the human and data webs. [online] available: http://www.w3.org/TR/xhtml-rfda-primer
[47] R. Khare (2006), Microformats: The next (small) thing on the semantic web? Internet Computing 10(1), 68-75
[48] B. Ostermaier, K. Romer, F. Mattern, M. Fahrmair, & W. kellerer (2010). A real-time search engine for the Web of Things. Proceedings of Internet of Things 2010, pp. 1-8