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

System Architecture Design

For the pSHIELD-project

Deliverable D2.3.2

Partners contributed to the work:

HAI, Greece
SE, Italy
UNIROMA1, Italy
ATHENA, Greece,
THYIA, Slovenia
ETH, Italy
SESM, Italy,
CS, Portugal,
CWIN, 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		
Andrea Morgagni	SE		
Vincenzo Suraci	UNIROMA1		
Kyriakos Stefanidis	ATHENA		
Athanasios Poulakidas	HAI		
Ljiljana Mijić	THYIA		
Spase Drakul	THYIA		
Gordana Mijic	THYIA		
Paolo Azzoni	ETH		
Emilio Bisbiglio	SESM		
João Cunha	SESM		
Przemyslaw Osocha	SESM		
Baldelli Renato	SE		
Andrea Fiaschetti	UNIROMA1		
Jose Verissimo	CS		
Reviewed by			
Name	**Company**		
Spase Drakul	THYIA		
Approved by			
Name	**Company**		
Josef Noll	MAS		

Modification History

Issue	Date	Description
Draft A	26.10.2011	Issue for comments, connection to D2.3.1
Issue 1	22.11.2011	SPD Requirements, Interfaces
Issue 2	24.11.2011	Overlay additions
Issue 3	01.12.2011	Network Interfaces and Review
Issue 4	09.12.2011	Final
Issue 5	10.01.2012	Approved
Issue 6		
Issue 7		

Contents

Figures

Tables

Glossary

AES	Advanced Encryption Standard
ADC	Analog Digital Converter
AOA	Agent Oriented Architecture
AP	Application Processor
ARF	Access Relay Function
ASIC	Application Specific Integrated Circuit
BGF	Border Gateway Function
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CI	Critical Infrastructure
CLA	Cross-Layer Architectures
CLD	Cross-Layer Design
CLASS	Cross-Layer Signalling Shortcuts
CL-pSSA	Cross-Layer pSHIELD System Architecture

CLIM	Cross-Layer Architecture and Interaction Module
CPU	Central Processing Unit
CR	Cognitive Radio
DAC	Digital Analog Converter
DAMA	Demand Assignment Multiple Access
DES	Data Encryption Standard
DLC	Data-Link Control
DMA	Direct Memory Access
DRA	Dynamic Resolution Adaptation
DRAM	Dynamic Random Access Memory
DRM	Digital Rights Managements
DSA	Digital Signature Algorithm
DSP	Digital Signal Processor
DVB-RCS	Digital Video Broadcasting - Return Channel via Satellite
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ED	Embedded Device
EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electromagnetic
ES	Embedded System
ESD	Embedded System Device
FA	Functional Architecture
FCC	Federal Communications Commission
FFT	Fast Fourier Transform
FLASH	Non-Volatile Computer Storage Chip
GW	Gateway
HW	Hardware
FIPS	Federal Information Processing Standards
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
HSM	Health Status Monitoring
I/O	Input/Output
IMAP	Internet Message Access Protocol
IP	Internet Protocol
ISA	Instruction Set Architecture
IPSec	Internet Protocol Security
IW	Intelligent Wagon
IWI	Intelligent Wagon Infrastructure
L2TF	Layer 2 Termination Function
LA	Layer Agents
LDAP	Lightweight Directory Access Protocol
M2M	Machine-to-Machine
MAC	Medium Access Control
MCU	Microcontroller
MEM	Volatile Memory

MGF	Media Gateway Function
MIPS	Million Instruction Per Second
ML	Middleware Layer
NeL	Network Layer
NF	Network Feature
MRFP	Media Resource Function Processor
NMP	Nano Micro/Personal
NMPS	Nano Micro/Personal Sensor
NoL	Node Layer
NVM	Non-Volatile Memory
OL	Overlay Layer
OS	Operating System
OSGi	Open Service Gateway Initiative
OSI	Open Systems Interconnection
P2P	Peer to Peer
PAP	Policy Administration Point
PBM	Policy Based Management
PC	Pervasive Computing or Personal Computer
PHY	Physical
PDA	Personal Digital Assistant
PEP	Policy Enforcement Point
PKC	Public Key Cryptography
PKI	Public Key Infrastructure
PM	Power Management
PMT	Policy Management Tool
pSFA	pSHIELD Functional Architecture
pSNA	pSHIELD Node Adapter
pS-NC	pSHIELD Node Capabilities
pSSA	pSHIELD System Architecture
QoS	Quality of Service
RAM	Random Access Memory
RFID	Radio Frequency Identification
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest Shamir & Adleman
SA	Security Agent
SCA	Side-Channel Attacks
SDR	Software Defined Radio
SGF	Signaling Gateway Function
SHA	Secure Hash Algorithm
SHSM	System Health Status Monitoring
SIP	Session Initiation Protocol
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SoC	System on Chip

SPD	Security Privacy Dependability
SPI	Serial Periphheral Bus
SPP	Special Purpose Processor
SRAM	Static Random Access Memory
SS	Stable Storage
SSL	Secure Sockets Layer
SW	Software
TCG	Trust Computing Group
TCP	Transmission Control Protocol
TPM	Trust Platform Module
UART	Universal Asynchronous Receiver/Transmitter
UWB	Ultra Wideband
VHDL	Very high speed integrated circuit Hardware Description Language
VoIP	Voice over IP
WiMAX	Worldwide Interoperability for Microwave Access
WRAN	Wireless Regional Area Network
WSN	Wireless Sensor Network

This Page is intentionally left blank

1 Executive Summary

D2.3 is a deliverable of pSHIELD WP2, “Scenarios, requirements and system design,” separated into internal intermediate deliverable D2.3.1, “Preliminary system architecture design” and public final deliverable D2.3.2, “System architecture design”. As denoted by its title, the main objective is to describe a formal and conceptual overall system architecture, to address Security, Privacy and Dependability (SPD) in the context of Embedded Systems (ESs) as “built in” rather than as “add-on” functionalities, proposing and perceiving with this strategy the first step towards SPD certification for future ESs. The methodology adopted concerns the gradual process and interaction with the major project topics, which form the framework for the architecture design, such as requirements, application scenario, metrics and technology development in the four layers described in the project. The latter, hierarchically ascending, Node, Network, Middleware and Overlay layers comprise the pSHIELD proposal, an alternative but integrative to classical layered OSI model structure. Therefore, the concluding complete architecture is presented through the definition of the four layers, the interfaces between them and the overall framework synthesized.

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 pSHIELD architecture composability relies on the so called SPD modules. Indeed the pSHIELD architecture is composed by a 'mosaic' of innovative SPD functionalities, each one of the considered layers. The pSHIELD architecture is able to derive application instantiations of the general framework, selecting statically (at design time) and dynamically (at runtime) the best SPD functionalities for achieving the required SPD levels. In particular, referring to the abovementioned layers, the SPD modules will implement the following functionalities:

- At node layer, intelligent hardware and firmware SPD
- At network layer, secure, trusted, dependable and efficient data transfer based on self-configuration, self-management, self-supervision and self-recovery
- At middleware layer, secure and efficient resource management, inter-operation among heterogeneous networks
- At overlay layer, composability

R&D for embedded security, intended as a system issue that must be solved at all abstraction levels (protocols, algorithms, architecture), will lead, in the framework of this task, to a coherent, composable and modular architecture for a flexible distribution of SPD information and functionalities between different ESs while supporting security and dependability characteristics.

This framework in D2.3.2 aims, at the one hand, to explore the minimum set of interdependencies between applications and architectures in an efficient way and to systematically classify those with respect to SPD. On the other hand, it aims to produce a composable architecture which will include most critical elements, thus covering most of the SPD requirements for all the applications. This approach is expected to produce a multi-layered architecture, where each layer consists of several hardware and software SPD modules (components), since it is imperative to take into account the need for composable security, privacy and dependability.

The resulting architecture has to be reconfigurable, offline, meaning that mechanisms should be provided to the designer for enabling/disabling nodes in order to tailor the overall system to his needs. Furthermore, fault diagnosis and fault recovery have to be addressed both in hardware and software layers.

Intra-layer and inter-layer interfaces should be defined in the system architecture to ensure the correct communication among the different SPD modules.

3 Terms, Definitions and Approaches

(1) Embedded System (ES) is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs.

(2) An Embedded System is a microprocessor based system that is embedded as a subsystem, in a larger system (which ***may or may not be a computer system***).

In general, "embedded system" is not a strictly definable term, as most systems have some elements of extensibility or programmability. For example, handheld computers share some elements with embedded systems such as the operating systems and microprocessors which power them, but they allow different applications to be loaded and peripherals to be connected. Moreover, even systems which do not expose programmability as a primary feature generally need to support software updates. On a continuum from "general purpose" to "embedded", large application systems will have subcomponents at most points even if the system as a whole is "designed to perform one or a few dedicated functions", and is thus appropriate to call "embedded".

An ES is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular function. Industrial machines, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys (as well as the more obvious cellular phone and PDA) are among the myriad possible hosts of an embedded system. For example, the following figure shows the essential parts of an ES:

- Microprocessor / DSP
- Sensors
- Converters (ADC and DAC)
- Actuators
- Memory (On-chip and Off chip)
- Communication path with the interacting environment

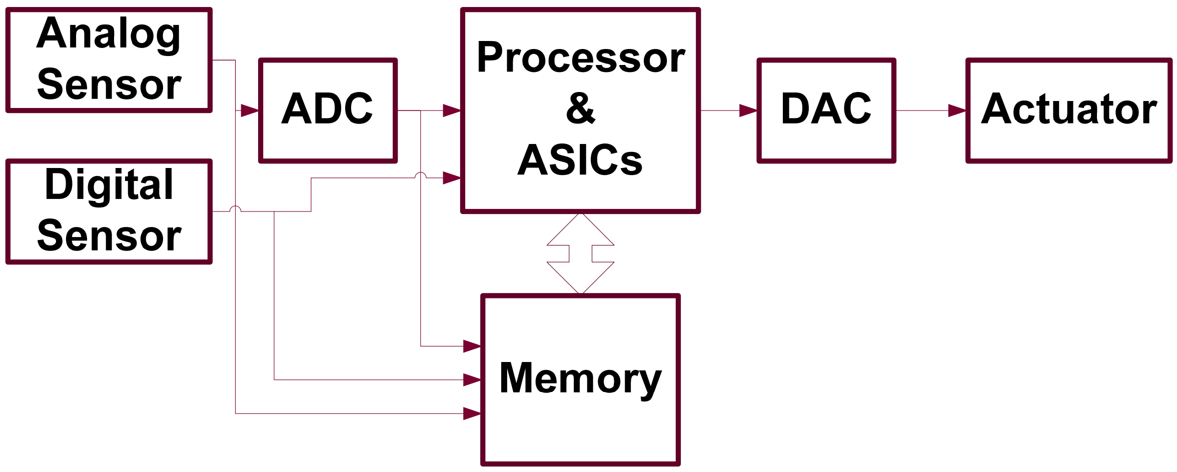


Figure 1 - A generic embedded system structure.

Embedded Device (ED) can be a small programmable chip that can be programmed to execute certain functions.

Heterogeneous system is a group of interacting, interrelated, or interdependent elements forming a complex whole.

Heterogeneous device integration designates technologies that can be integrated on one platform device.

Heterogeneous network is a network connecting computers and other devices with different operating systems and/or protocols.

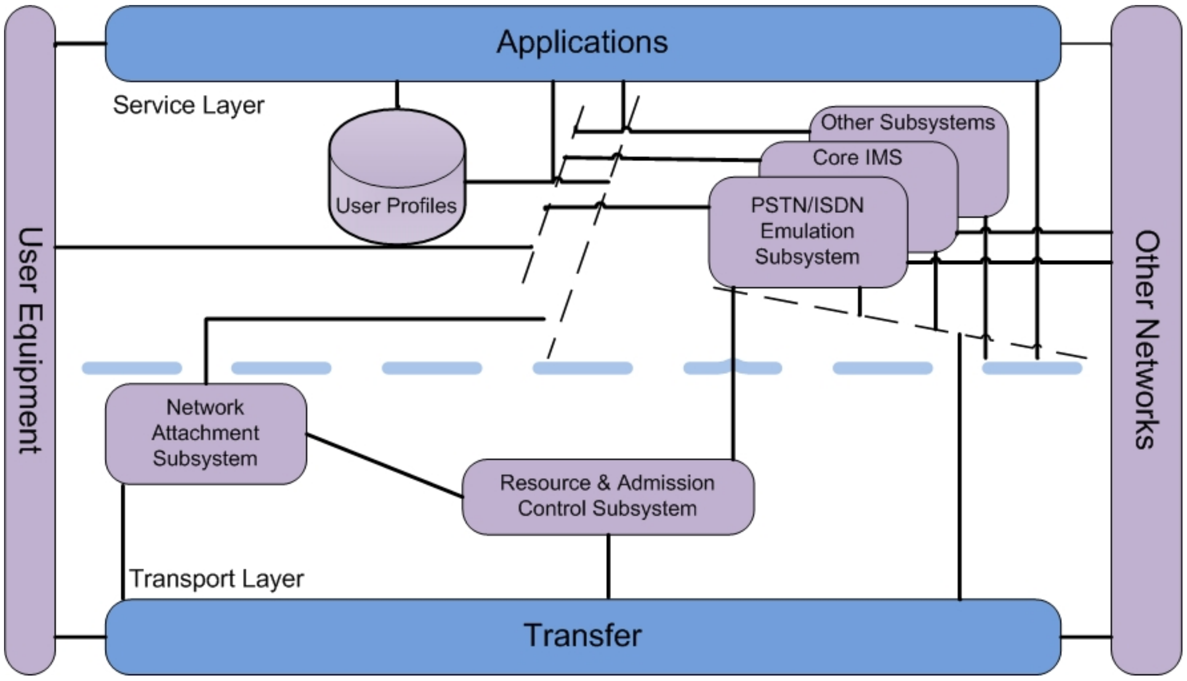
Architecture: The architecture of a system defines its basic components and important concepts and describes the relationships among them.

Functional architecture (FA): The functional architecture can be viewed as the set of basic information processing capabilities available to an information processing system. The functional architecture is comprised of a set of primitive operations or functions. This means that these basic functions cannot be explained by being further decomposed into less complex ("smaller") sub-functions.

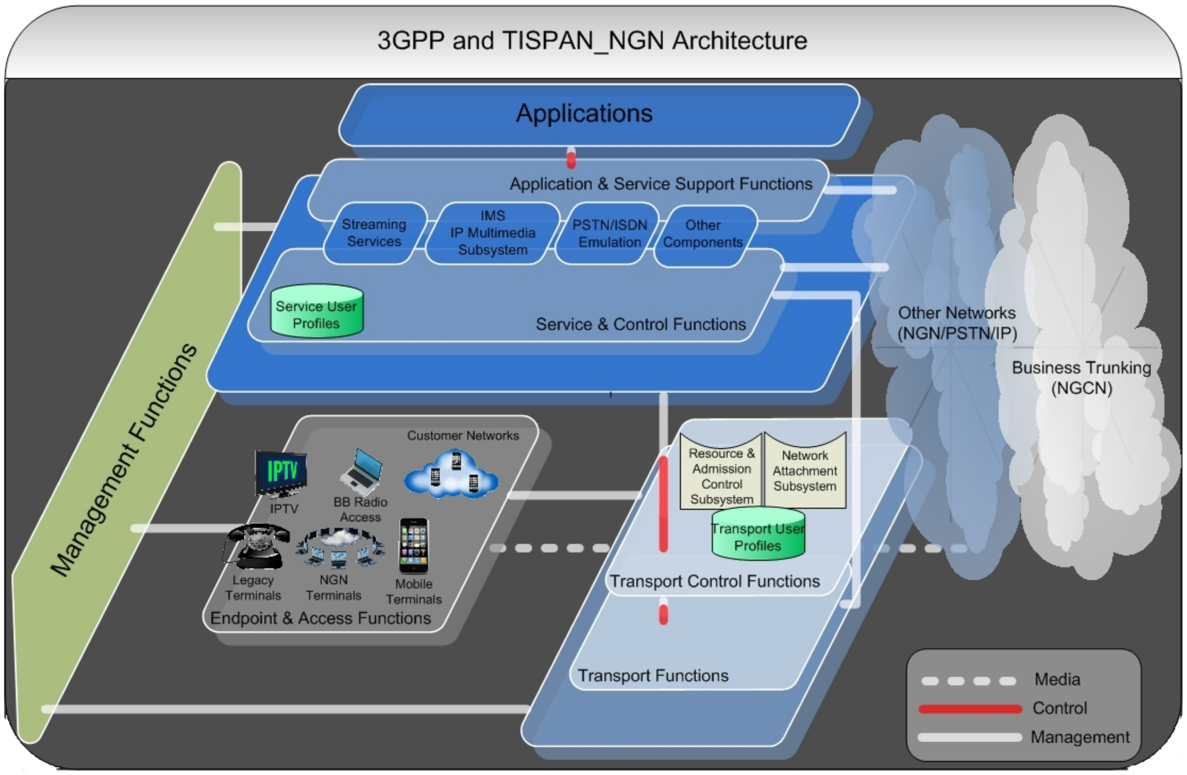
The functional architecture is constructed from an analysis of different functional requirements as deduced from different use cases.

Example 1:

Figure 2 shows one example of the TISPAN NGN functional architecture.



a)



b)

Figure 2 – a) Overview of the functional architecture of TISPAN NGN release 2, b) layered view of the TSPAN NGN architecture.

The main scope of the above functional architecture is to identify a set of functional blocks, for example: Media Gateway Function (MGF); Border Gateway Function (BGF); Access Relay Function (ARF); Signaling Gateway Function (SGF); Media Resource Function Processor (MRFP); Layer 2 Termination Function (L2TF), etc. Each subsystem is specified as a set of functional entities and related interfaces. As a result implementers may choose to combine functional entities where this makes sense in the context of the business models, services and capabilities being supported. Where functional entities are combined the interface between them is internal, hidden and un-testable.

Example 2:

The Web system architectures have a technical architecture that is divided in functional and information architecture. For both the information architecture and the functional architecture, various abstraction levels exist, which include the business architecture, logical architecture, physical architecture and implementation architecture:

- *Business Architecture*: The business architecture represents business processes, policies and procedures, workflows and user interactions. This is used to guide the design of other more technical related architecture layers
- *Logical Architecture*: The logical architecture defines at a high level the structure of the system to be developed. The elements in this layer are logical concepts, instead of concrete or physical software components

- *Physical Architecture*: The physical architecture further defines the technical solution at a detailed level. Some design decisions, such as the selection of content storage product, can be represented in this layer
- *Implementation Architecture*: The implementation architecture specifies system composition and interconnections

pSHILED Functional Architecture (pSFA) is composed by four functional layers: node, network, middleware and overlay, which represent a set of four functional sub-systems that are specified by its set of elements, functional entities and interfaces.

Node Layer (NoL) of pSFA: node layer is composed of Intelligent ES HW/SW Platform and have different kinds of Intelligent ES Nodes: nano node, micro/personal node, power node, and Dependable Self-X Crypto Technologies. This layer is composed of standalone and/or connected NoL elements like sensors and actuators, which perform smart transmission.

The NoL is a layer composed by physical nodes (i.e. hardware). Each node is a generic Embedded System (for example a sensor, an actuator, a transmitter etc.).

Network Layer (NeL) of pSFA is a heterogeneous layer composed by a common set of protocols, procedures, algorithms and communication technologies that allow the communication between two or more nodes.

Middleware Layer (ML) of pSFA is the software layer that provides the basic functionalities to use the underlying networks of embedded systems (like service discovery and composition) as well as some security functionalities (like accounting or access control). This layer, being software, is installed on the nodes.

Overlay Layer (OL) of pSFA is a logical vertical layer that collects (directly or indirectly) semantic information coming from the Node, Network and Middleware layers and uses them to compute the adequate actions (if any) that ensure the desired level of SPD. It is a software routine running at middleware and/or application level.

3.1 Cross-Layer / Cross-Overlay Architecture Definitions

3.1.1 Introduction

The purpose of this chapter is to give an overview on the cross-layer architectures (CLAs) proposed recently by the research community and to address in details SPD issues for overall pSHIELD System Architecture (pSSA).

In wireless and mobile networking envisioned for pSHIELD System Architecture (pSSA), difficult environmental conditions are a permanent challenge, resulting in a demand for cross-layer optimizations. There is also a need to further increase flexibility of the network. Therefore, we believe cross-layer architectures should adapt themselves to these changing conditions, just as they adapt the network stack, devices, and applications.

The network protocol stacks are logically organized in layers. These layers are strictly separated and the cross-layer functionality between them is restricted by determined interfaces, which in effect only allow passing packets up and down the stack. In principle, all these layers have been designed to fulfil their functionality without interaction across the layers. History shows that this works well in wired and static environments. Popularity and success of wireless networks and highly mobile nodes is currently dominating in new development and research activities. In order to adapt to the rapidly and frequently changing network conditions under those circumstances, a more sophisticated interaction between protocols than in a traditional layered architecture is desirable. The existing solutions are not able to dynamically change which of these to use and how to use them, i.e., the adaptation, re-parameterization and addition of cross-layer optimizations during runtime. Moreover, customization of optimizations in existing frameworks is often cumbersome and complicated, if it can be done at all.

Cross-layer architectures diverge from the existent network design approaches, where each layer of the protocol stack operates independently and the data between the successive layers is exchanged in a very strict and systematic manner. There are several advantages of a layered approach since modularity, robustness and ease of design are effortlessly achieved. The modularity that the layers provide allows for potential arbitrary combination of protocols and the maintainability is being improved as new versions of a protocol can be inserted without having to alter the rest of the network stack. However the properties of the different layers have substantial interdependencies and a modularized design may be suboptimal with regards to performance especially in satellite and mobile wireless environments, where the communication channels and traffic patterns are more unpredictable than in wired-line networks.

There has been much talk about cross-layer design for wireless communication networks lately. It has been argued repeatedly that layer boundaries, as specified in the layered architectures, are not suitable for wireless communications and performance gains can be made by giving up strict layering to do cross-layer design [1], [2].

This section discusses a communication methodology involving node cooperation which, while demonstrating a new opportunity created by wireless networks, significantly challenges the layered architecture.

Cross-layer design touches not just communications and networking, but is also intimately connected to concepts related to communications architecture. Layered architectures have served to make the protocol design activity systematic and modular. Potential performance gains can always motivate a designer to not follow the layered architectures and do cross-layer design. But cross-layer design cannot be seen as an end itself.

This section presents both, a state-of-the-art (SoA) of the ongoing work and platforms over which new research can be built [3]-[20]. In this project we will discuss specific cross-layer design proposals and new alternatives. But mainly, we encourage a more holistic treatment of cross-layer design itself.

We therefore propose a Cross-Layer pSHIELD System Architecture (CL-pSSA) with the following properties:

- Signalling between an arbitrary amount of layers and system components
- Extensibility of the architecture and adaptability of optimizations at runtime
- High usability for cross-layer developers via an abstract description language for optimization rules

There also exist several cross-layer architectures facilitating signalling across all layers, i.e. any-to-any layer signalling. For example, Cross-Layer Signalling Shortcuts (CLASS) enables direct signalling between all layers by message passing [8].

3.1.2 Description of Cross-Layer Architecture

3.1.2.1 A definition for Cross-Layer Design (CLD)

A layered architecture, like the seven layer Open Systems Interconnection (OSI) model [3], defines a hierarchy of services to be provided by the individual layers. The services at the layers are realized by designing protocols for the individual layers, which can be implemented on the target platform to obtain a complete system.

At the protocol design phase, the designer has two choices. Protocols can be designed by respecting the rules of the original architecture. In the case of the layered OSI reference model, this would mean designing protocols such that they only make use of the services at the lower layers and not be concerned about the details of how the service is being provided. It also implies that the protocols would not need any interfaces that are not present in the reference architecture.

Alternatively, protocols can be designed by violating the reference architecture. Since the reference architectures in communication and networking have traditionally been layered, its violation is generally termed as cross-layer design.

1.1.1.1 A taxonomy of CLD

In recent times, a large number of cross-layer designs have been proposed. A classification based on the layers that are coupled by the different proposals can be found in [4]. In this section, we classify the existing cross-layer design proposals according to the kind of architectural violations they represent. Two points should be mentioned here. Firstly, our coverage of the cross-layer design proposals is meant to be representative and not exhaustive. Secondly, the reference architecture we assume is motivated from the "best of both worlds" five-layer model proposed in [5]. Thus, we assume that the reference architecture has the application layer, the transport layer, the network layer, the link layer which comprises the data-link control (DLC) and medium access control (MAC) sub-layers [3], and the physical layer—with all the layers performing their generally understood functionalities.

We identify the following architectural violations:

1) Creation of new interfaces (Figure 3: A, B, C)
2) Merging of adjacent layers (Figure 3: D)
3) Design coupling without new interfaces (Figure 3: E)
4) Vertical calibration across layers (Figure 3: F)

A. Creation of new interfaces

Several cross-layer designs require creation of new interfaces between the layers. These can further be divided into three categories depending on the direction of information flow along the new interfaces:

1) Upwards: From lower layer(s) to a higher layer
2) Downwards: From higher layer(s) to a lower layer
3) Back and forth: Iterative flow between the higher and lower layer

We now discuss the three sub-categories in more detail and point out the relevant examples.

1) *Upward information flow*: A higher layer protocol that requires some information from the lower layer(s) at runtime results in the creation of a new interface from the lower layer(s) to the higher layer, as shown in Figure 3 A
2) *Downward information flow*: Some cross-layer design proposals rely on setting parameters on the lower layer of the stack at run-time using a direct interface from some higher layer, as

illustrated in Figure 3 B. Such downward flow of information is termed as Hints in [6]. As an example, the applications can inform the link layer about their delay requirement and the link layer can then treat packets from the delay sensitive applications with priority [7].

3) *Back and forth information flow*: Two layers, performing different tasks, can collaborate with each other at run-time. Often, this manifests in an iterative loop between the two layers, with information flowing back-and-forth between the layers, as highlighted in Figure 3 C.

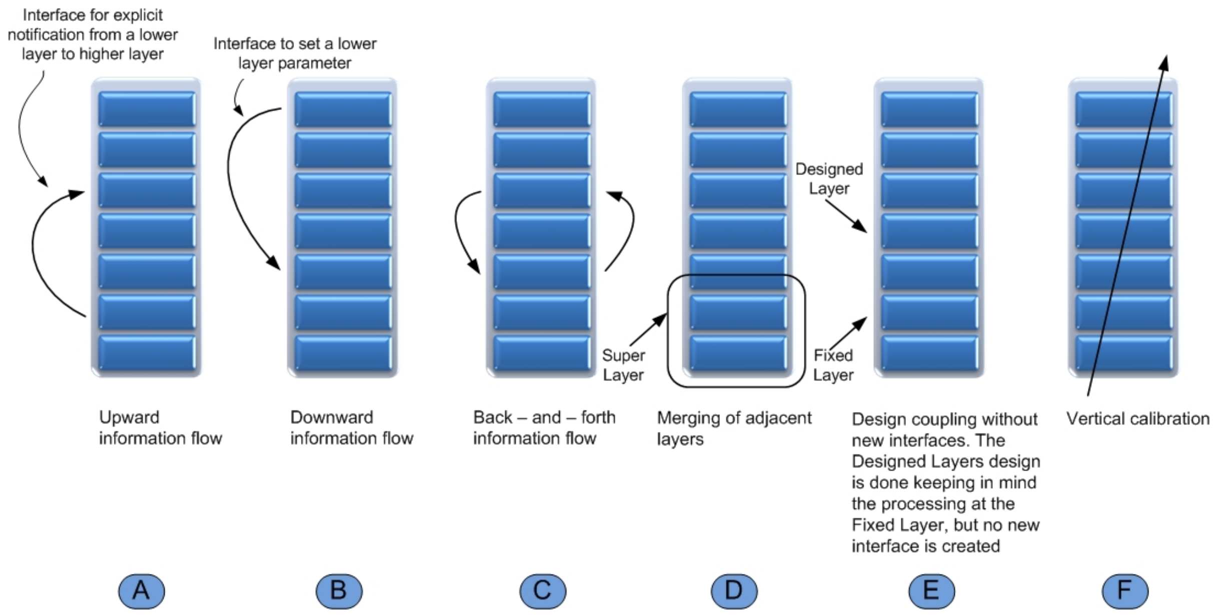

Interface for explicit notification from a lower layer to higher layer
Interface to set a lower layer parameter
Designed Layer
Super Layer
Fixed Layer
Upward information flow
Downward information flow
Back – and – forth information flow
Merging of adjacent layers
Design coupling without new interfaces. The Designed Layers design is done keeping in mind the processing at the Fixed Layer, but no new interface is created
Vertical calibration
A
B
C
D
E
F

Figure 3 - Different kinds of cross-layer design proposals (boxes represent the protocol layers).

3.1.2.2 Existing and new CLAs

3.1.2.2.1 A classic CLA

Cross-layer design proposals that we looked at in Section 3.1.2.2 demonstrate the violation of layered architectures at the protocol design phase itself. Hence, a question that naturally comes up is, “Can there be architectures that are general enough such that protocols for wireless networks can be designed without violating them?” In fact, this is a complicated question. Determining what the new architectures should look like requires the study of not only the performance issues from a communication or networking viewpoint, but also an understanding of the implementation related issues. Nevertheless, some preliminary proposals have been made in the literature.
These can be put into two categories:

1) Allowing the layers to communicate with each other (Figure 4 A)
2) A shared database across the layers (Figure 4 B)

A. Allowing the layers to communicate

A straightforward way to allow information sharing between the layers is to allow them to communicate with each other, as depicted schematically in Figure 4 A. Practically speaking this means making the variables at one layer visible to the other layers at run-time. Notice that under strictly layered architectures, every layer manages its own variables and its variables are of no concern to other layers.

There are many ways in which the layers can communicate with one another. For instance, protocol headers may be used to allow flow of information between the layers. Alternatively, the extra “inter-layer” information could be treated as internal packets. The work in [8] presents a comparative study of several such proposals and goes on to present another such proposal, namely, the Cross-layer signalling shortcuts (CLASS).

CLASS allows any two layers to communicate directly with one another. Similarly, the Hints and Notifications proposal discussed in [6] makes network layer the hub of inter-layer communication. These proposals are appealing in the case where just a few cross-layer information exchanges are to be implemented in systems that were originally designed in conformance with the layered architectures. In that case, one can conceivably “punch” a few holes in the stack while still keeping it tractable. However, in general, when variables and internal states from the different layers are to be shared between the different layers as prescribed by such proposals, a number of implementation issues relating to managing shared memory spaces between the layers may need to be resolved.

B. *Shared database across the layers*

The other architecture proposal recommends a common database that can be accessed by all the layers, as illustrated in Figure 4 B. See for instance references [9] and [4]. In one sense, the common database is like a new layer, providing the service of storage/retrieval of information to all the layers.

The shared database approach is particularly well suited to vertical optimizations. An optimization program can interface with the different layers at once through the shared database. The main issue here is the design of the interactions between the different layers and the shared database.

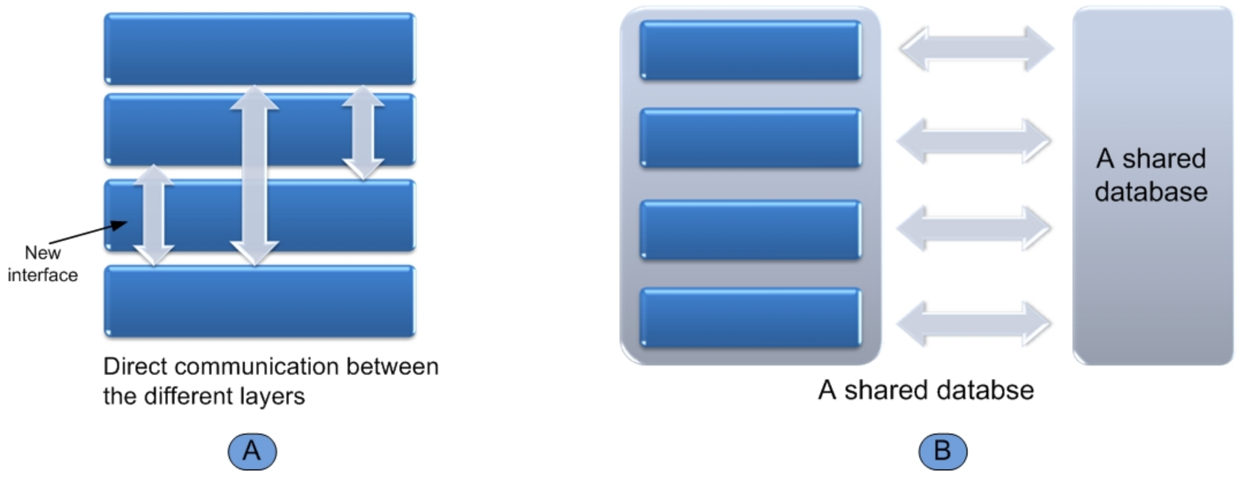


Figure 4 - Proposals for architectural blueprints for wireless communications.

3.1.2.2.2 New CLA

3.1.2.2.2.1 Cross-Layer Interaction Model

The cross-layer architectures proposed in the literature do not address all the design goals of interoperability, rapid prototyping, maintainability, portability and efficiency. Starting from this consideration, the goal of this work is to provide a generic framework for building and organizing a cross-layer interactions model (CLIM) which could serve as a unified and simple way to implement cross-layer optimizations (see Figure 5). When using CLIM, the concept of NF (Network Feature) should also be introduced. A NF is either a functional service that can be provided to the end-user (e.g. QoS), or a network component whose operations/configurations are supposed to be critical in terms of performance, efficiency or services, at the system level or for the user satisfaction. Basically, cross-layer interactions may be local to or distant within a network node.

In many cases only two elementary NF are involved in the adaptation and interaction (one source NF, one target NF). In some other cases multiple (local or distant) entities could participate. Local communications between protocols of non-neighbour layers are done through a local interface that must be defined.

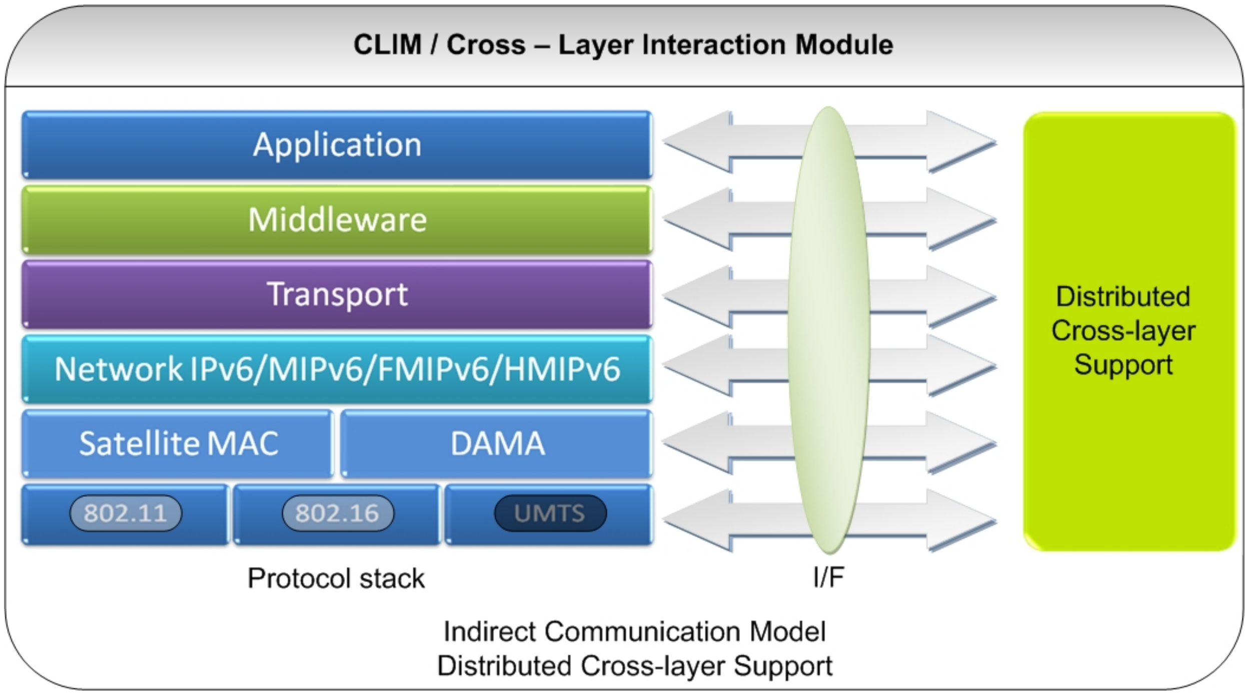

CLIM / Cross – Layer Interaction Module
Application
Middleware
Transport
Network IPv6/MIPv6/FMIPv6/HMIPv6
Satellite MAC
DAMA
802.11
802.16
UMTS
Distributed Cross-layer Support
Protocol stack
I/F
Indirect Communication Model
Distributed Cross-layer Support

Figure 5 - Proposed Cross-Layer Interaction Model

The cross-layer architecture and interaction module CLIM has two main components: 1) QoS and Resource Management and 2) Mobility Management. The QoS and Resource Management component includes:

- SIP and MAC cross-layer interactions used to support the interworking between WiMAX and DVB-RCS and multimedia QoS-aware application
- Transport layer and MAC cross-layer interaction (i.e. the interaction between TCP PEP (Performance Enhancing Proxy) and DAMA (Demand Assignment Multiple Access) in the MAC)

designed to optimize the way in which the available resources is used taking into account QoS mapping at the MAC layer and enabling data to be sent to the lower layers at the speed at which the MAC layer queues are emptied (flow-control)

- IP and MAC scheduling interaction implemented in a way that can fully take advantage of QoS capabilities offered by the satellite system
- MAC and Physical layer interaction between DRA and DAMA as information in the frame constitution

The Mobility Management component utilizes a slightly modified model (CLIM-m) based on the more generic CLIM architecture together with ideas from [17], [18]. The mobility management modules include:

- algorithms for handover prediction and decision algorithms for fast handover with handover preparation, handover coordination and optimization algorithms for best performance, and
- information to decide the appropriate time to initiate and execute the handover procedures.

The proposed cross-layer platform for mobility management specifies a Cross-layer Manager consisting of a Link Information Manager and a Handover Manager connected to Layer Agents (LA). The Handover Manager communicates with multiple protocol layers via the LAs which capture specified parameters in each layer when the respective values are changing beyond a certain threshold. These data are reported to the Handover Manager to collect sufficient information for the Handover Decision Unit.

3.1.2.3 Challenges

There are additionally several open questions, some of which cannot be addressed from a performance viewpoint alone and require a consideration of architectural concerns too. For example:

- What should be the role of the physical layer in wireless networks?
- Is the conventional view of the network—that of a collection of point-to-point links—appropriate for wireless networks?
- How do the different cross-layer design proposals co-exist with one another?
- Will a given cross-layer design idea possibly stifle innovation in the future?
- What are the cross-layer designs that will have the most significant impact on network performance, and hence should be most closely focused on?
- Has a given design proposal been made with a thorough knowledge of the effect of the interactions between the parameters at different layers on network performance?
- Under what network and environmental conditions can a particular cross-layer design proposal be invoked?
- Can the mechanisms/interfaces used to share information between the layers be standardized?
- How do we make sure that the new architectures allow innovative usage of the wireless medium that we are likely to see in the future?

We will investigate at some of these issues in greater detail.

3.1.2.4 Description of Layers

3.1.2.4.1 Physical Layer

In wired networks, the role of the physical layer has been rather small—that of sending and receiving packets when required to do so from the higher layers. As we have seen, advances in the signal processing at the physical layer can allow it to play a bigger role in wireless networks. Consider, for instance, multi-packet reception capability at the physical layer.

3.1.2.4.2 MAC Layer

The MAC layer's functionality is intimately connected to the network layer (and hence to the rest of the stack), we see that signal processing advances at the physical layer promise to have a significant impact on all aspects networking protocol design. Cross-layer designs relying on advanced signal processing at the physical layer are an interesting research ground for the future.

3.1.2.4.3 Cross-Layer Principles

The solutions for cross-layer adaptation seek to enhance the performance of the system by jointly optimizing the performance of single or multiple cross-layers. The uncertainty is to what extent the layered architecture needs to be modified in order to introduce co-operations among protocols belonging to different layers. At one end, solutions based on triggers between the layers implement interdependencies between protocols while maintaining compatibility with strict layering. A full cross-layer design represents the other extreme; this implies introducing stack-wide layers' interdependencies that enable the optimization of each protocol's performance by exploiting the full knowledge of the network status abstracted at different layers of the protocol stack.

However, in a multiple-objective optimization scenario, care should be taken to avoid undesirable (and unpredictable) interactions across parameters in various layers, leading to conflicts or even loops between the layers. There also exists a design trade-off between the multiple optimization goals and the effect of the increased processing and interactions to achieve these goals. Unfortunately by doing cross-layer design in an undisciplined way, it is likely to end with a poorly structured system and to greatly increase the complexity of an already complex system [10][11].

Figure 6 illustrates the main ideas of cross-layer adaptation and optimization in a hybrid terrestrial and satellite network. Cross-layer optimization may be implemented locally (intra-node) or globally (inter-node). A number of cross-layer methods and architectures have been proposed in the literature [12][13][14] [15][16]. They all share some common features and diverge notably in the way the cross-layer principle is implemented, on what kind of application they focus, in the capacity the architecture has and where the actual adaptation intelligence is located.

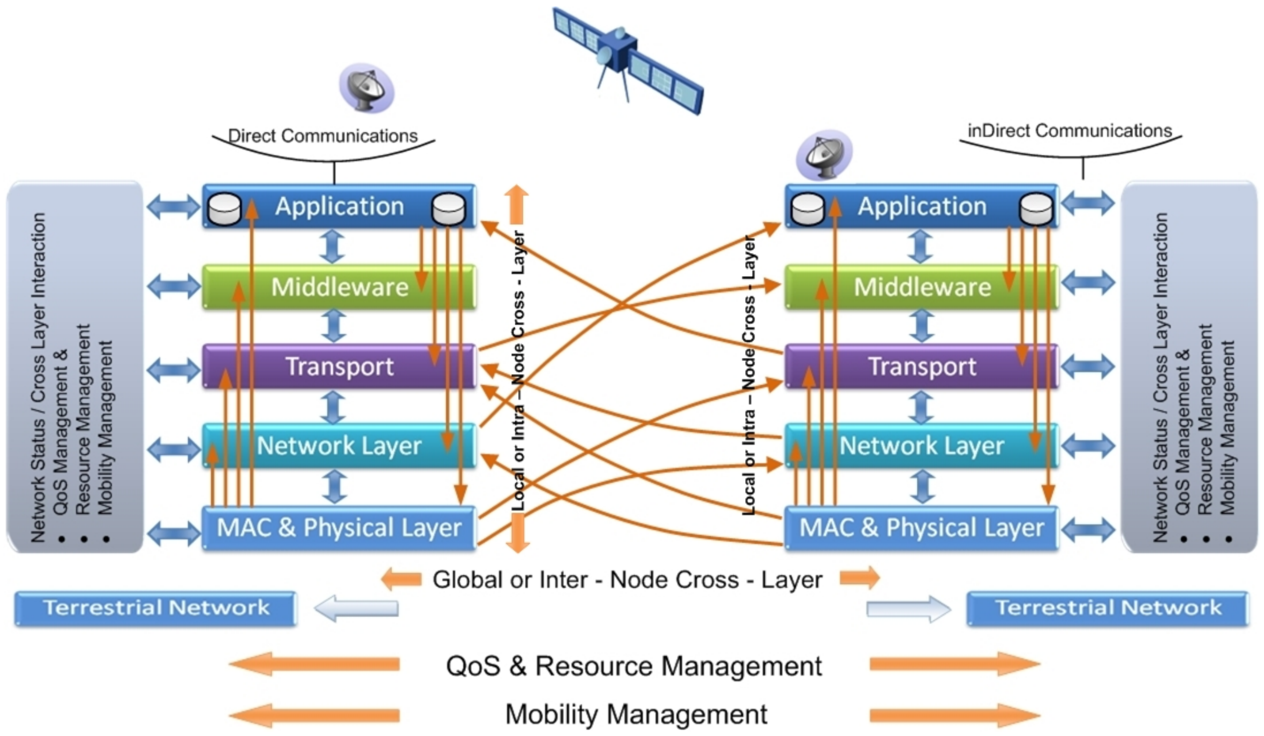


Figure 6 - Cross-layer adaptation and optimization in satellite network.

These architectures mostly fit out into one of the two categories: direct or explicit cross-layer communications and indirect or implicit cross-layer communications via a common entity, see Figure 7.

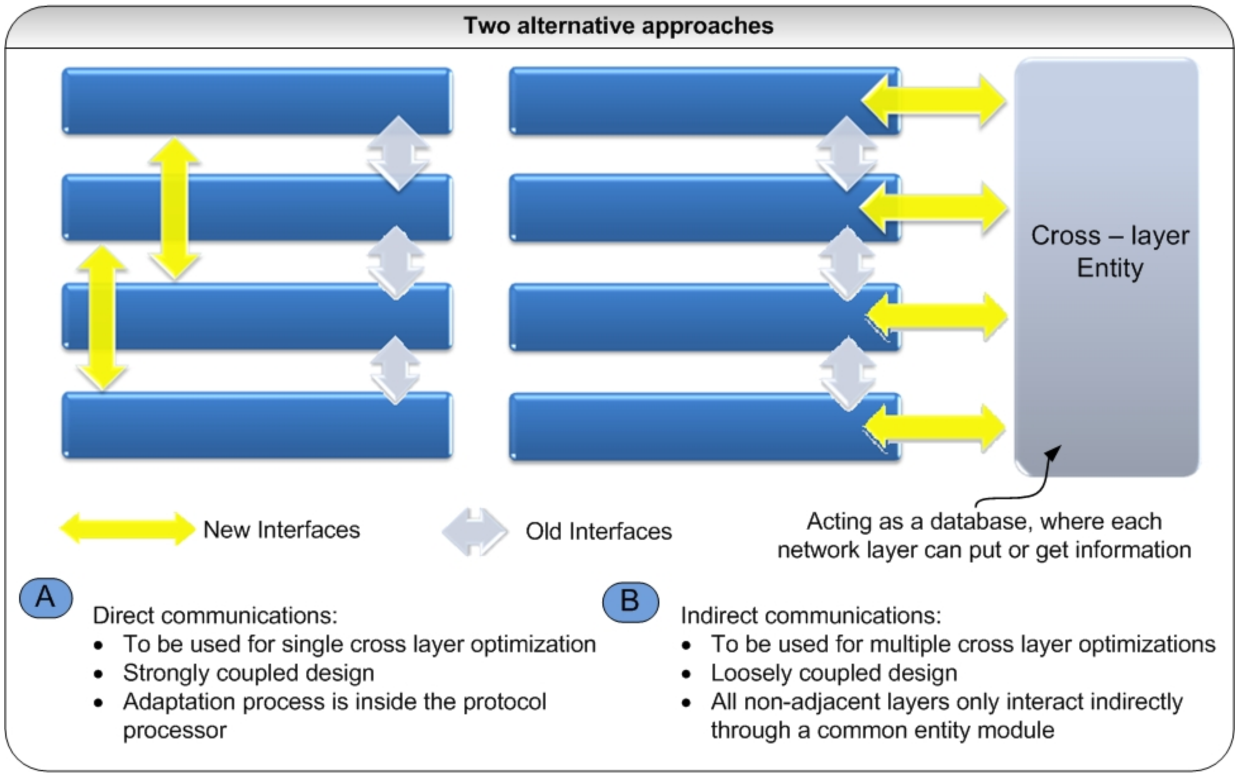

Two alternative approaches
Cross – layer Entity
New Interfaces
Old Interfaces
Acting as a database, where each network layer can put or get information
A
Direct communications:
• To be used for single cross layer optimization
• Strongly coupled design
• Adaptation process is inside the protocol processor
B
Indirect communications:
• To be used for multiple cross layer optimizations
• Loosely coupled design
• All non-adjacent layers only interact indirectly through a common entity module

Figure 7 - Cross-layer architectures for indirect and direct communications.

The first category, direct communications, should be used when only a single cross layer optimization is planned. The second category, indirect communications, is realized with a common cross-layer entity or cross-layer manager, which acts as a mediator between the layers. The cross-layer entity includes a network status component of the stack that interfaces the different layers between themselves and acts as a database where each network layer can put or get information. This architecture should be used for multiple cross layer optimizations.

3.1.3 The Overlay as a Cross-Layer Security Architecture for Security, Privacy and Dependability

As we have seen in the previous section, a Cross-Layer Architecture is a design paradigm that could improve the performance of a generic system by simply allowing information exchange among the layers. This approach has been widely described for Telecommunication environment, however for the purpose of the pSHIELD project we need to adapt it to a different context by addressing two main issues:

i) while classical Cross-Layer is applied to the seven, well known, ISO/OSI layers, pSHIELD Cross-Layer is applied to three heterogeneous and complex layers named Node (the hardware), Network (the interconnection between nodes) and Middleware (the software services located between the hardware and the applications providing the core SPD functionalities)

ii) while classical Cross-Layer aims at optimizing telecommunication quality of service performances (like bandwidth, transmission delays and so on), the pSHIELD Cross-Layer is in charge of addressing Security, Privacy and Dependability to bring them to a reference, desired value. With Security, Privacy and Dependability we refer to a wide set of functionalities relevant for the system like Authentication, Cryptography, Authorization, Key management, Auditing, etc. For any possible threat, pSHIELD acts as a SPD overlay, applying a functional-cross-layer approach, as depicted below:

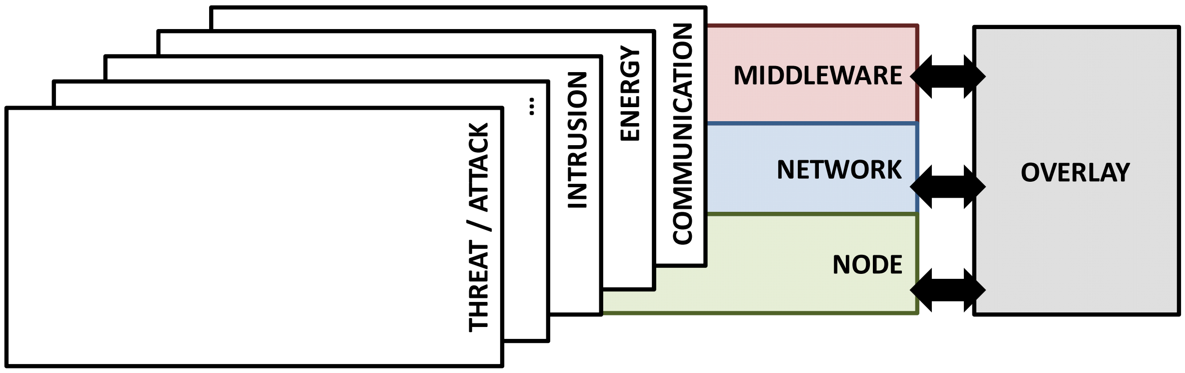


Figure 8 – pSHIELD Overlay.

Given a threat or an attack (such as intrusion, tampering, energy shortage, communication fault, etc.) it potentially impacts all middleware, network and node layers. To face this threats and attacks, traditional approaches are to defend each single functionality or each single layer, without any coordination with the rest of the involved system. To ensure such an holistic approach it is necessary a Cross-Layer element to serve as a glue between the different domains (node, network and middleware). The Overlay monitors and controls the SPD level of the whole system.

In order to simplify the description, in the unfolding of the document we will refer to the "pSHIELD Cross-Layer Architecture for Security, Privacy and Dependability" with the word "Overlay Layer" or pSHIELD Overlay".

Since there are many SPD functionalities positively affected by the introduction of an Overlay layer, most of them depending on the application scenario, it is not possible to provide in advance an extensive, quantitative description of the advantages for each SPD functionality. However in the following sections some examples are provided to better understand the Overlay impact on improving the SPD level of a given system of embedded systems.

3.1.3.1 Security

The Overlay approach could improve the security of an interconnected set of Embedded Systems by leveraging the basic security functionalities of each hardware and/or software components. For example if an encryption algorithm is required to protect the output of a node, the overlay could decide to perform it directly at node level (by activating the adequate chip and producing an already encrypted output) as well as at network level (by ciphering the information while transmitting it over a secured channel) or at middleware level (by ciphering the information stored in memory before sending it). Of course these solutions are equivalent if and only if they all fit the application needs.

1.1.1.2 Dependability

An intuitive example of Dependability functionality that could be guaranteed by the pSHIELD overlay could be the power consumption of the system. If we consider, for example, a monitoring system composed by different nodes, communication protocols and software components, the Overlay could dynamically choose the less expensive (in terms of energy) configuration of devices or transmission protocols and active services in order to maximize the life of the battery and consequently the dependability of the system itself.

3.2 Node Layer Definitions

3.2.1 Nano and Micro/Personal Node

A **pSHIELD Node** is an Embedded System Device (ESD). When a Legacy ESD equipped with several legacy node capabilities will be used in the pSHIELD network it requires a pSHIELD Node Adapter (pSNA). A pSHIELD node is deployed as a hardware/software platform, encompassing intrinsic, innovative SPD functionalities, providing proper services to the other pSHIELD networks and middleware adapters to enable the pSHIELD composability and consequently the desired system SPD. There are three kinds of **pSHIELD node** deploying each different configuration of Node Layer SPD functionalities of the pSHIELD framework, and comprising different types of complexity: **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.

The technological 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 method for designing pSHIELD NMP Nodes is twofold:

1. To design completely **new NMP nodes** that are **complaint with the pSHIELD system** design.
2. To keep legacy node technologies as they are compliant with their standards, developed for many applications including those that are targeted in pSHIELD, which means to assume 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 **legacy NMP node** with physical layer and protocol stack composed of existing and new SPD technologies added by pSNA. As result of this integration a new types of networked SPD ESDs will be created. pSHIELD and new SPD ESDs will compose a heterogeneous SPD network infrastructure too.

Developing a NMP node as integrated NMP-SPD Node of a Legacy NMP node and pSNA we obtain a composable pSHIELD Node. It means that it has all of the desired SPD functionalities and services for the pSHIELD application scenario selected. Additionally to that, the pSHIELD Node keeps some of the desired functionalities of a standardised sensor technology with **additional SPD features** that make it composable into the pSHIELD framework. The architectural design of the pSHIELD Nodes will relay on the ISO/IEC 9126 standard that has 6 top level characteristics:

- functionality,
- reliability,
- usability,

- efficiency,
- maintainability and
- portability.

The architectural design of the NMP nodes is not an easy architectural task since it requires considering many different constraints at the same times. Some of these constraints can converge in the same direction but some of them will be divergent and in the opposite directions. To cope with this challenge architectural design, as shown in section 4, the pSHIELD ESD use two approaches:

- network approach and
- functional approach.

The **network approach** constrained the architectural system design from network point of view. This approach should guarantee that all NMP nodes are part of a pSHIELD-SPD network that can be easily integrated with standard IP-based network like GSM, UMTS, etc. In other words it means that an SPD network is implementable and interoperable with standard networks to comply the main business cases of the application scenarios.

The **functional approach** constrained architectural design from the SPD requirements point of view and it is related to the node, network, middleware and overlay layers. The real innovation of pSHIELD is the introduction of the **Overlay** that makes the two approaches converge.

Nano- and micro- sensor technology encompasses the family of devices in the dimension scales of nanometers and micrometers respectively. They are simple machines with limited capabilities and resources that include sensing, simple computations, small memory and data transmission in short distances. Biological sensors, inside the human body and sensors integrated in things, such as books or keys are two example types of nano- and micro- sensing. The rise of these two technologies (and combinational schemes of them) is expected to have great impact in many aspects of social life, such as homeland security or environmental protection.

Nano- and micro- sensor technology encompasses the family of devices in the dimension scales of nanometers and micrometers respectively. They are simple machines with limited capabilities and resources that include sensing, simple computations, small memory and data transmission in short distances. Biological sensors, inside the human body and sensors integrated in things, such as books or keys are two example types of nano- and micro- sensing. The rise of these two technologies (and combinational schemes of them) is expected to have great impact in many aspects of social life, such as homeland security or environmental protection.

Such devices are often integrated in larger units to form nano- or micro-machines dedicated to sensing or actuation functionalities. Interconnections of these machines create networks able to serve applications in larger areas, expanding the degree of range and complexity. Further, these networks can communicate with other networks or connect on Internet based applications to make the distribution of information even more widespread.

Nano-engineering is one of the new eras of technological challenge and the research is on going. In this context, the communication techniques of these networks are still immaturely standardized. It is usually not crystal clear how these small devices communicate. Furthermore, a lot depend on the topology, the components and network architecture of the specific application. It is self-evident that the development of nanonetworks implies the arrangement of a wide set of parameters (just like every communication network), including:

- Architectural framework (sensors, routers, gateways)
- Frequency band of operation
- Channel modelling (path-loss, noise, bandwidth and capacity)
- Modulation
- Protocols (MAC addressing, routing, service discovery)

Therefore, a brief technical summary of the pSHIELD nano- and micro- platform is considered useful, in contrast to further theoretical definitions.

The pilot demonstrator of pSHIELD uses the Sun SPOT sensor platform as micro node. Sun SPOT is a useful platform for developing and prototyping application for sensor network and embedded system. Sun SPOT is suitable for application areas such as robotics, surveillance and tracking.

3.2.1.1 Technology Description

The SPD goal for the NMPS node prototypes is to take in consideration the following design constrains:

I. For RT scenario, which belongs also to critical infrastructure (CI), 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.

Development platform should have 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.

3.2.1.2 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.

3.2.1.2.1 Microcontroller/Microprocessor comparison

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. Table below shows the comparison of the MCUs.

MCU	RAM (kB)	FLASH (kB)	Active (mA)	Sleep (µA)	Sensor Nodes
Atmega128 (Atmel)	4	128	8	20	DSY25, EberNet, BT node, Iris, MicaZ, Mica2
AT91SAM7128 (Atmel)	32	128	30	10	Evaluated
Atmega644/V (Atmel)	4	64	0.4	0.1	TelG Mote
STM32W108B* STMicroelectronics	8	128	6@12MHz	<1	pSHIELD NMPS node
PIC Modern (Microchip)	4	60	2.2	1	CIT Sensor node, Particle 2/29, GWnode
80C-51 (Philips)	2	60	15	3	ECO, MITes
MSP430F14x (TI)	2	60	1.5	1	Telos, BSN node, Pluto
MSP430F16x (TI)	10	48	2	1	eyesIFXv2, Tmote Sky

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

For example, that Atmega644P/V has the lowest consumption for both active and sleep modes. 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! Further information about this chip is provided in the Appendix. Since pSHIELD was targeted as pilot project for 12 months duration it was not possible to implement SPD features on this chip. It will be furthermore investigated in the following up project nSHIELD, which is a three year project.

3.2.1.2.2 IEEE 802.15.4 chips comparison

For WSNs the selection of the radio is a critical for NMPS nodes, because the performance should not be evaluated for a individual NMPS node. The application requirements define what type of radio is needed. A wideband radio operating at 2.4 GHz and comply with IEEE 802.15.4 standard offer advantages that are important for the pSHIELD scenario. There are several radio modules available in the markets that are in compliant with IEEE802.15.4 standard. Most of the module differences lie on its power profile, device interface and additional features. Several IEEE802.15.4 compliant radio from Atmel, Chipcon, Microchip and MaxStream are listed Table below. When very high data rate are required (depends of the application requirement) AT86RF231 is the best choice. XBEE module from MaxStream has 50 mW output power and range from 40 m - 1.6 km. The module also provides a complete solution including the antenna. Other radio chip requires a careful design of an external antenna. The USART device interface is very easy to configure and XBEE has two modes of operation which are transparent and API mode.

	Atmel AT86RF231	**Chipcon CC2420**	**Microchip MRF24J0MA**	**MaxStream XBEE**
Data rate (kbit/s)	250, 500, 1000, 2000	250	250	250
Rx power (mA)	12.3	19.7	19	50
Tx power (mA/dBm)	14/+3	17.4/0	23/0	45
Power down (μA)	0.02	1	2	<10
Turn on time (ms)	<0.4	0.58	Not available	Not available
Device interface	SPI	SPI	SPI	USART
IEEE 802.15.4 HW support	FCS, CCA, RSSI, ED and LQI	RSSA, LQI	RSSA, LQI	RSSI
Antenna	External	External	Integrated PCB	Integrated Whip

Hardware components

The main units are Sunspot devices with embedded sensors and base station. Each Sunspot has a so-called eSPOT with battery, while the base station is not equipped with battery and must be powered from the host computer via a USB cable. The Sunspot does not need to run any operating system, it needs only JVM that runs on bare metal and executes directly out of flash memory. Stack-boards are composed of specific sensors and actuators such as accelerometers, light sensors and temperature sensors. The hardware components of a sensor board are as follows:

- 180MHz for 32-bit ARM920T core processor with 512K RAM and 4M Flash, runs on Squawk
- 2,4GHz based IEEE 802.15.4 radio (radio ChipCon TI CC2420) which is integrated in the antenna
- USB interface for connecting to a host computer
- 3.7V battery (720 mAh), Sleep mode (32 uA)
- 3 axis accelerometer (2G/6G)
- tri-color LEDs, 2 push-buttons control switches
- digital I/O pins, 6 analog inputs, 4 digital outputs

Integrated sensors

- Temperature Sensor: Chip-type is ADT7411 sensor that measures temperature with ADC. ADC is integrated into eDemo, and can measure temperatures between -40℃ to +125℃
- Accelerometer sensor: 3-axis accelerometer of the type LIS3L02AQ, designed by ST Micro Systems and located in eDemo Board. This sensor can measure the x-axis, y-axis and z-axis in

the direction up and down with the value either ±2G or ±6G. When the Sunspot is at rest, it measures x = y = 0 and z = 1G

- Light sensor is of the type TPS851, designed by Toshiba. The sensor can measure the voltage between 0.1V (dark) - 4.3V (light), and converts the voltage to the brightness of Luminance (lx) 3

Software components: SUN SPOT JVM

Squawk is open source and has been written in the Java programming language. It is a virtual machine, and is a highly portable Java VM. The advantage of Squawk is that it can run on bare metal instead of being run on top of the operating system. This means that applications can be isolated and be treated as application objects. This allows multiple applications running on the same virtual machine. Squawk also supports CLDC 1.1 that facilitates connectivity to mobile phones.

3.2.2 Power Node

The Power Node is a rugged embedded system, providing high computing power, optimally designed in terms of dimensions, weight, power consumption and capable to work in harsh environmental conditions. The reference application context is defence/aerospace ground mobile and airborne environments, addressing manned and unmanned applications where reliable high performance computing is required.

The Power Node is a pSHIELD SPD Embedded System Device. It is a physical component that offers native SPD features at different abstraction levels: hardware, firmware, network and operating system. The native support of SPD features and the high level of standard adopted, in terms of hardware architecture and operating system, make it ready to host middleware and overlay services.

The node offers low level native SPD capabilities, which are available as hardware components, hardware interfaces, firmware functionalities and network functionalities.

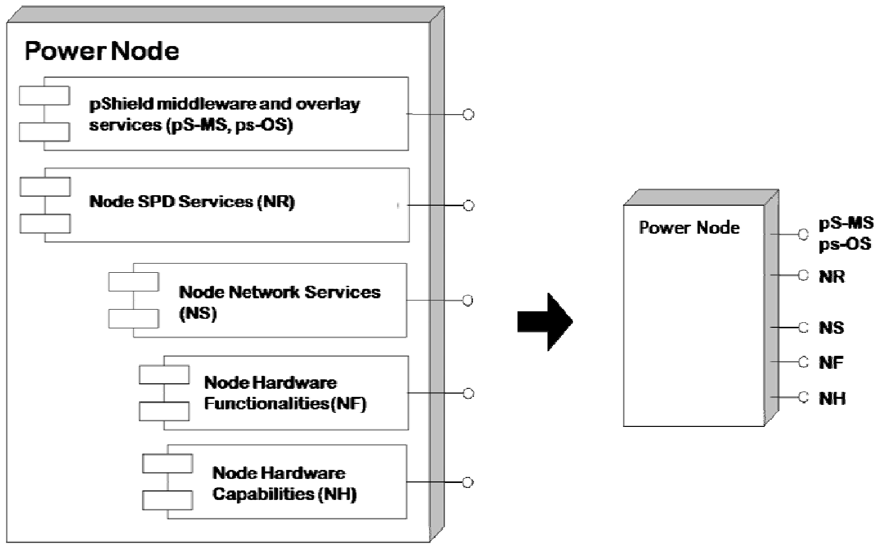

Power Node
pShield middleware and overlay services (pS-MS, ps-OS)
Node SPD Services (NR)
Node Network Services (NS)
Node Hardware Functionalities(NF)
Node Hardware Capabilities (NH)
Power Node
pS-MS
ps-OS
NR
NS
NF
NH

Figure 9 – Power Node pSHIELD component.

The node hardware capabilities are focused on security, dependability and composability. These features are provided by the architecture of the node, by the hardware components available on it, by its hardware interfaces, by the firmware and by the compliance with MIL standards. The features include:

- components which can be reconfigured (i.e. FPGA) and interfaces that allows reconfiguration at system level (redundant power supply, node redundancy, spare node, etc.)
- set of FPGA SPD core blocks
- firmware that supports remote reconfiguration of BIOS, FPGA images, etc.
- power supply monitoring and protection unit
- monitoring devices that allow to control continuously the status of the node
- programmable I/O and network interface
- BMC embedded unit
- MIL standard to ensure that the node is capable to work in critical environmental conditions

The node hardware functionalities intend to support security, privacy and dependability. These functionalities are based on the capabilities and features offered by the hardware available on the Power Node board. The functionalities include:

- reconfigurablity functionalities for components, interfaces and firmware
- functionalities for security monitoring (through a dedicated FPGA core logic)
- cryptographic functionalities (through a dedicated core of the FPGA or using Intel AES-NI technology)
- functionalities to monitor continuously working parameters, performance and status of node
- composability functionalities that allow to implement networks of Power Nodes, increasing fault tolerance, computing power, security and dependability

The network services are oriented to increase privacy and dependability. These services are intended mainly to provide some Power Node hardware functionalities remotely. The network services include:

- services for remote reconfiguration of BIOS and FPGA
- SNMP services
- services for remote self-test and performance monitoring
- privacy and login management services

The node SPD services rely on the hardware capabilities and functionalities, on the network services and on the operating system, offering a set of more abstract services that can be used by the middleware and overlay layers. The set of services includes:

- FPGA and BIOS reconfiguration services
- privacy and user control services
- composability services
- status and performance monitoring services;
- services for the management of the topology of Power Nodes Network

The Power Node has been conceived to host entire parts of the pSHIELD middleware and of the pSHIELD overlay, in particular when the requirements in terms of computing power, hardware composability and reliability are important. In this context, it acts as a computing and reasoning node, more than a data acquisition node. The middleware and overlay services that can be hosted depend on the application context and the role of the parts themselves. From an operating system and application point of view the Power Node can be seen as a very powerful rugged personal computer therefore, any part of the pSHIELD middleware and overlay that runs on a PC may run also on the Power Node.

3.2.3 Cryptography Technologies

This section presents in a summarized way the most relevant cryptography technologies related terms and definitions used in pSHIELD project. It is divided in the following sections: attacks on cryptosystems, attacks on protocols, asymmetric and symmetric cryptography, message authentication codes and key management.

Elliptic curve cryptography (ECC) is becoming a powerful cryptographic scheme. Because of its efficiency and security is a good alternative to cryptosystems, like RSA and DSA, not just in constrained devices, but also on powerful computers. ECC is very important in the field of low-resource devices such as smart cards and Radio Frequency Identification (RFID) devices because of the significant improvements in terms of speed and memory compared to traditional cryptographic primitives (e.g. RSA). Memory is one of the most expensive resources in the design of embedded systems which encourages the use of ECC on such platforms. Security, implementation and performance of ECC applications on various mobile devices have been examined and it can be concluded that ECC is the most suitable PKC scheme for use in a constrained environment.

More and more electronic transactions for mobile devices are implemented on Internet or wireless networks. In electronic transactions, remote client authentication in insecure channel is an important issue. For example, when one client wants to login a remote server and access its services, such as on-line shopping and pay-TV, both the client and the server must authenticate the identity with each other for the fair transaction.

The remote client authentication can be implemented by the traditional public-key cryptography. The computation ability and battery capacity of mobile devices are limited, so traditional PKC, in which the computation of modular exponentiation is needed, cannot be used in mobile devices. Elliptic curve cryptosystem (ECC), compared with other public-key cryptography, has significant advantages like smaller key sizes, faster computations. Thus, ECC-based authentication protocols are more suitable for mobile devices than other cryptosystem. However, like other public-key cryptography, ECC also needs a public key infrastructure (PKI) to maintain the certificates for users' public keys. When the number of users is increased, PKI needs a large storage space to store users' public keys and certificates. In addition, users need additional computations to verify the other's certificate in these protocols.

A WSN is a wireless ad-hoc network consisting of resource-constrained sensoring devices (limited energy source, low communication bandwidth, small computational power) and one or more base stations. The base stations are more powerful and collect the data gathered by the sensor nodes so it can be analyzed. Routing is accomplished by the nodes themselves as any ad hoc network through hop-by-hop forwarding of data. Common WSN applications range from battlefield exploration and emergency rescue operations to surveillance and environmental protection.

Security and cryptography on WSNs meet several open problems even though several years of intense research. Given the limited computational power and the resource-constrained nature of sensoring devices, the deployment of cryptography in sensor networks is a difficult task. In D3.2 is presented the implementations of elliptic curve cryptography in the tiny sensor nodes. Design goals for a sensor platform is to develop optimizations specifically:

(i) the cost of memory addressing;
(ii) the cost of memory instructions;
(iii) the limited flexibility of bitwise shift instructions.

D3.2 work presents efficient implementations for arithmetic of binary field algorithms such as squaring, multiplication, modular reduction and inversion at two different security levels. These implementations take into account the characteristics of the target platform. The implementation of field multiplication and modular reduction algorithms focuses on the reduction of memory accesses and appears as the fastest result for this platform.

3.2.3.1 Attacks on Cryptosystems

There are a number of techniques that have been used in the past to exploit weaknesses of some cryptographic algorithms and are currently used as basic evaluation criteria for new algorithms. The common aim of these attacks is to reveal partially or entirely the information encrypted in intercepted messages, or to extract some information internal to the encryption process (without initially knowing any secrets). They include:

- Brute force attack - traversing the entire encryption key space in order to learn the encryption key
- Dictionary attack - related to the brute force attack in that a set of keywords are used as possible values of the encryption key (or a pass phrase)
- Chosen cipher text attack - obtaining information about a secret decryption key by submitting a range of cipher texts to decrypt
- Adaptive chosen cipher text attack - a version of chosen cipher text attack in which the attacker interactively selects subsequent cipher texts based on the results of decryption of the previous ones
- Cipher text-only attack - the attacker has access to a limited set of cipher texts
- Known plain text attack - the attacker has access to a number of cipher texts together with the corresponding plain texts
- Chosen plain text attack - the attacker can encrypt an arbitrary set of chosen plain texts
- Adaptive chosen plain text attack - like above, but the attacker chooses subsequent plain text for encryption based on the previous results
- Related-key attack - the attacker has access to encryption of a plain text under several different keys whose exact values may not be known but which are somehow mathematically related

In addition to these general attack methods, there is also a range of more general cryptanalytic techniques that may be used to study the properties of ciphers. They include frequency analysis, differential cryptanalysis, linear cryptanalysis, statistical cryptanalysis and mod-n cryptanalysis. Finally, there are also attacks on hashing functions (e.g., birthday attack) that aim at finding collisions in hash functions or attacks on random number generators that exploit a generator’s statistical weaknesses to simplify breaking a cipher that uses it.

3.2.3.2 Attacks on Protocols

Communication and security protocols can be attacked in a number of ways by intercepting and inserting messages in the communication channel. These attacks are even easier to perform in wireless networks since there might be little difficulty in accessing the channel, unless a more sophisticated technology such as direct-sequence spread spectrum (DSSS) or frequency hopping is used.

- Replay attack - resending of some captured messages in order to confuse the protocol or to exploit some of its weaknesses
- Wormhole attack - a form of a replay attack that uses a low-latency and long-range transmission link to intercept communications in one part of the network and then to reproduce them in another network region, for example, with the goal of authenticating the attacker
- Man-in-the-middle attack - the attacker intercepts all communications from a node A, modifies them and sends to a node B in such a way that both A and B have the illusion of direct communication with each other

- Bit flipping attack - selectively flipping bits in intercepted messages in order to achieve desired protocol behaviour, for example, to route traffic to different recipients or to change the message type
- Attack on key distribution protocols - preventing or intercepting key distribution in the network might severely affect the entire safety infrastructure of the system
- Routing protocol attacks - the attacker may influence the contents of routing tables of some network nodes or even to introduce corrupt nodes to affect communication in the network

3.2.3.3 Asymmetric Cryptography

Asymmetric cryptography, also known as public key cryptography, is based on the disposition of two types of keys, a public key and a private key, that are used in the cryptographic operations. Intuitively, the public key is made available by a given entity to potential senders while the private key is kept hidden by that entity. A message sent to an X receiver should be encrypted by X's public key where X can later decrypt it using its private key.

There are mainly three well-known types of asymmetric cryptography algorithms (Eisenbarth & Kumar, 2007): Elliptic Curve Cryptography (ECC), Rivest Shamir Adleman (RSA) and EL-Gamal. Depending on the target application and scenario specifications, implementations of the aforementioned approaches can be in software, hardware or a co-design of both.

3.2.3.4 Symmetric Cryptography

Symmetric ciphers use the same key or a pair of trivially-related keys (e.g., one is a linear transformation of the other) for both encryption and decryption of messages. Historically, symmetric ciphers precede their asymmetric counterparts and although less versatile in their applications, they continue to be widely used due to the fact that they are typically several orders of magnitude faster, as well as, they can be implemented more efficiently. The main downside of symmetric key cryptography is the need to establish a secure communication channel for key exchange between the communicating parties before the actual communications can begin. As a result, asymmetric (public key) cryptography is often used to exchange symmetric session keys between the two parties and then to use a symmetric cipher to encrypt all subsequent communications.

Symmetric ciphers can be grouped into two broad categories: **stream ciphers** and **block ciphers**. The former combine a pseudo-random bit sequence with the plaintext (typically an XOR) and, thus, operate on individual bits or bytes of the plaintext, while the latter use fixed-size blocks of plaintext. Stream ciphers are typically faster and simpler to implement than block ciphers, both in software and in hardware, and are better suited for encryption of transmissions of streams of large amounts of data (e.g., video streams). However, stream ciphers have been reported to have serious security vulnerabilities when not used carefully. In particular, keys should never be reused otherwise the plaintext can be easily recovered.

Block ciphers use fixed-size blocks of plaintext, typically of 128 bits and transform them in a sequence of operations, called rounds. Encryption of messages longer than the block size is done using a mode of operation, i.e. a technique of partitioning the plaintext into a sequence of blocks and then chaining their encryption to construct the cipher text of the entire message. Encryption of plaintexts smaller than the block size is done using a padding scheme.

3.2.3.5 Message Authentication Codes

The ability to create a unique and non-forgeable digest of a message is of great practical importance. In particular, message authentication can be implemented by directly linking the sender's identity to the message's contents in form of a message authentication code (MAC). There are two general ways of implementing MACs: using cryptographic hash functions and running block ciphers running special modes such as, for example, cipher block chaining (CBC).

3.2.3.6 Key Management

A key management scheme is an integral part of any deployed security system. Whether the cryptographic approach followed is symmetric or asymmetric, the role of an efficient key management scheme is vital. Such a scheme is affected by the system's architecture, device classes, deployment environment, potential attacks and other factors. For example, a key management scheme for a secure WSN needs to deal with the limitations of such a system in terms of nodes computational, storage and energy constraints in addition to expensive wireless communication. Essentially, key management comprises key pre-distribution approaches and other schemes dependent on the nature of the network. In all cases, certain basic operations should be supported such as key addition, revocation and renewal.

3.3 Network Layer Definitions

3.3.1 General Network Layer description

Network Layer is responsible for delivering data packets of variable length between hosts belonging to different networks. Implementation depends on application environment and possible collaborative networks. Generally, it may include connectionless communication, hosts addressing and message forwarding, with IP and IP-based protocols being the most popular technology patterns. Network Layer design goes in accordance with MAC specification and may involve topics, such as, selection of the *Internet Protocol Suite*, Mobile IP for IPv4/IPv6 and Security Management, where IPsec is an open standard security scheme for authentication and encryption of IP packets.

The pSHIELD Network Layer is charged with routing, multi-fold connectivity tasks and trusted data transfer among system components. Representative (for the scope of pSHIELD) topology of participating nodes and their homogeneous or not "islands", form the network's structure. The offered communication capabilities and overall network functions are presented. The interactions and communication with other layers and modules should be described through well defined interfaces. A conceptual and modular Network Layer architecture is designed through a methodology that encompasses all the critical features described already as the focus of the current study. Network Layer multi-technology architectural design is based on a series of "key" factors, adjusted in the view of pSHIELD networking needs. The most prominent of these factors (to be taken under consideration) are the following:

- 6 key concepts of pSHIELD
- Network requirements, phrased in deliverable D2.1
- Selection of Nodes
- Security, Privacy and Dependability functions and features
- Application Scenario
- Software stack
- Reference Architecture
- Connectivity and Trusted Routing
- Technology status
- Metrics
- Interactions and Interfaces, Cross Layer Architecture

- Robustness and Composability in the general system framework
- Commitment to the Technical Annex

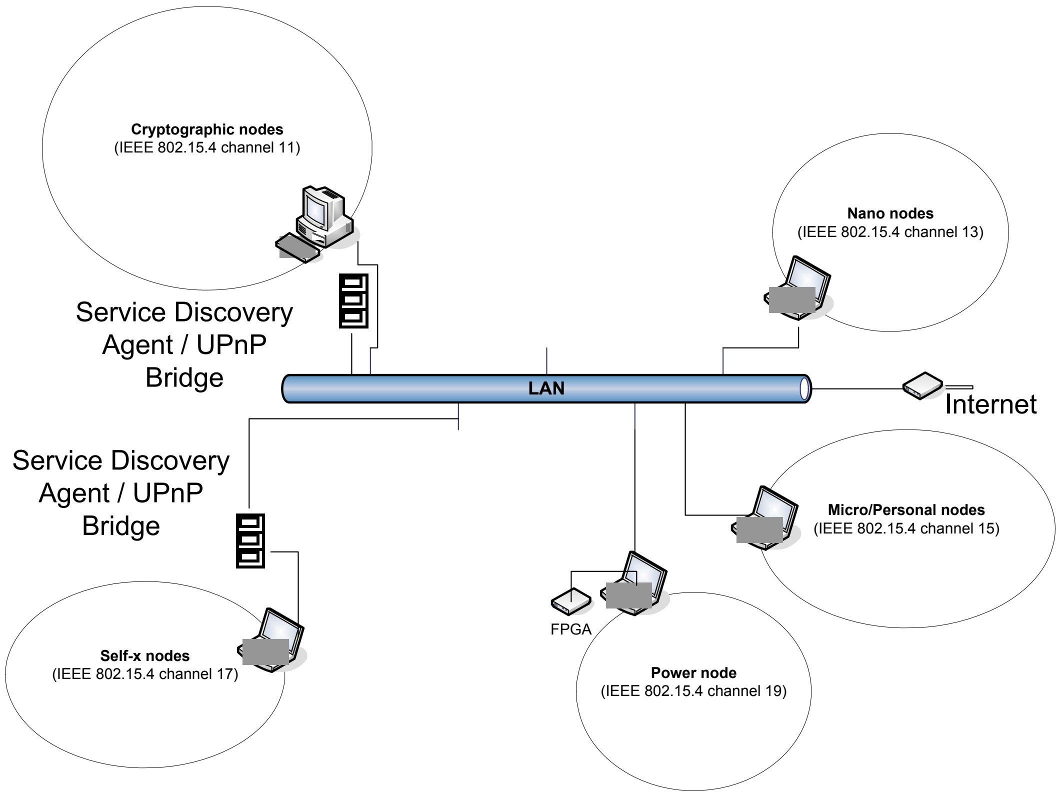

Cryptographic nodes
(IEEE 802.15.4 channel 11)
Nano nodes
(IEEE 802.15.4 channel 13)
Service Discovery Agent / UPnP Bridge
LAN
Internet
Service Discovery Agent / UPnP Bridge
Micro/Personal nodes
(IEEE 802.15.4 channel 15)
FPGA
Self-x nodes
(IEEE 802.15.4 channel 17)
Power node
(IEEE 802.15.4 channel 19)

Figure 10 - Network Layer: paradigm architecture.

3.3.2 Software Defined Radio

SDR platform will be used to provide smart SPD driven transmission. It is a radio communication software system, implemented on software (e.g., on embedded devices), meant to replace system components, such as amplifiers, filters, mixers, modulators, originally implemented in hardware. It can receive and transmit a variety of different radio waveforms, based on the software used. It can, also, be integrated easily with hardware security modules. Significant advantages can be derived from this alternative realization of a radio communication system:

- Variable network parameters can be set straightforwardly and on-line, whereas in hardware implementation reconfigurations are, frequently, cumbersome or need to be conducted manually
- The communication itself can be changed, providing the possibility of access to different networks (e.g., GSM and Wi-Fi) without having to switch between different devices
- Signal processing can be managed by a general purpose processor, providing a radio that can receive and transmit heterogeneous waveforms (depending on the specific software used)
- SDR is flexible in overcoming problems of limited spectrum availability
- Addition of an SDR node in a mesh network increases its capacity and reduces energy consumed per node

Smart transmission techniques by pSHIELD Network Layer will rely on waveform-agile implementations of SDR. The two basic elements which model an SDR system are a computing unit and an RF front end. In pSHIELD Architecture, SDR in cooperation with Cognitive Radio (CR), will be used to derive from SPD modules the corresponding desired communication and security functionalities.

3.3.3 Cognitive Radio

Cognitive Radio is a radio communication standard, able to change its transmission and reception parameters, according to conclusions deduced from sensing the frequency spectrum. The objective is to detect the unused spectrum and utilize it to set up radio communication. CR scouts the environment and adapts to user needs, without neither involving the user in this procedure, nor interfering with the host network, while conforming to FCC rules. As its inspirator Joseph Mitola III described, the motive behind CR is to render users aware of available radio resources, in order to serve their communication needs, without obstructing these resources' allocation and normal function of the whole network. CR is itself a network node performing (concisely) the following actions, in the procedure of setting successfully a radio communication:

- Detection of free frequency spectrum
- Creation of a channel to use this spectrum
- Function without interfering with native devices

And more analytically, the main functions and methods of CRs could include:

- Spectrum Sensing
    - ✓ Transmitter detection: CR detect *signals* from specific *transmitters* in specific *spectrum*
    - ✓ Cooperative detection: information from multiple CRs is used for detection
    - ✓ Interference based detection
- Spectrum Management
    - ✓ Spectrum Analysis
    - ✓ Spectrum Decision
    - ✓ Spectrum Pooling: allocation of resources between users
    - ✓ Spectrum Leasing
    - ✓ Spectrum Sharing: a stack of spectrum is given to users (similar to MAC channel addressing)
    - ✓ Negotiated Spectrum Use
    - ✓ Opportunistic Spectrum Use
    - ✓ Dynamic Spectrum Access
- Spectrum Mobility
    - ✓ A CR user exchanges its operation frequency

Although primary observations concern frequency spectrum usage, other factors can be inspected also, such as user behaviour, channel conditions, network state and link performance. Subsequently the CR may decide to alter communication settings, from which the most usual are waveform, protocol, frequency and power, to meet the required level of QoS.

CR uses unlicensed frequency bands or alternatively, fairly used licensed ones. Its success led FCC to consider open further bands for unlicensed use. IEEE 802.22 is a standard for CR air interface (PHY/MAC), belonging to WRAN communication protocols. It is developed for Cognitive Radio techniques, to exploit geographically unused spectrum. Additionally, IEEE P1900 is a standard committee and group focusing on the development of standards dealing with spectrum management.

Having a brief view on the architectural schemes of SDR/CR, we can add to prerequisite blocks (apart from the radio front end and the computer system we saw in previous paragraph) a control unit, capable of making decisions following spectrum sensing and concerning methods of spectrum management. Variability of options appears in the computing unit selection:

a) General purpose processor
b) Application specific processors and integrated circuits
c) FPGAs for reconfigurable computing
d) Embedded Systems

The last category shows that Embedded Systems and SDR/CR architectures fit quite well and direct us to pSHIELD Network Layer application techniques. Reversely, speed of ESs, together with low energy requirements, rather small programming effort and the specific nature of tasks they perform make SDR/CR schemes a suitable communication model. Also, due to the nature of ESs' Operating Systems, SDR/CR devices can have, comfortably, quick reset and reconfigurability processing times, without the demand of lengthy off-line periods. The cost of the combination SRD/CR with ESs could be very reasonable, allowing us to seamlessly improve network performance in application domains. Cognitive Radio, for example, is an emerging radio approach in emergency communications, to face issues of vital importance, such as the usual network congestion during the peak of a crisis or difficulties in intercommunication between authorities resulting from heterogeneous communication protocols and systems.

In chapter 6, System Design, SPD Driven Transmission and Trusted Connectivity are examined further, as functions of pSHIELD Network Layer, provided by the appliance of SDR/CR architecture.

3.4 Middleware Layer Definitions

The Middleware layer is seen, from a pSHIELD perspective, as a pure software-based layer playing a key role to interface the Node and Network layer to the Overlay layer. The Middleware layer is shown in the below figure highlighted in grey.

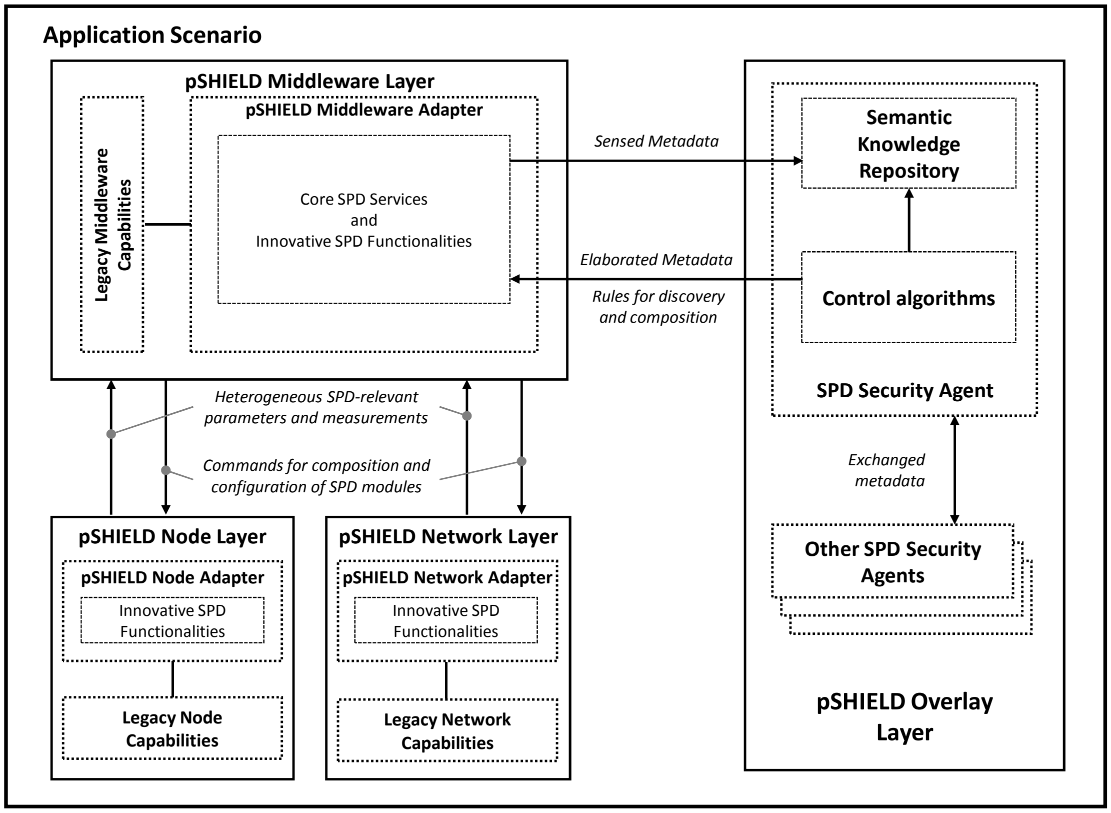

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 11 – pSHIELD Middleware layer.

Given an application scenario, the pSHIELD Middleware layer is in charge to get heterogeneous SPD-relevant parameters and measurements from the Node and Network layer adapters. This information, jointly with the ones gathered from the Middleware layer, are semantically enriched and sent to the Overlay layer as sensed metadata. Once the Overlay layer elaborates these information as inputs of proper control algorithms, it sends back to the Middleware layer the best rules to discover and compose the available SPD middleware, network and node resources. Indeed the discovery and composition services are part of the Core SPD Services provided by the pSHIELD Middleware layer. The Core SPD Services are detailed in the following section. Once activated, the Core SPD Services elaborate the proper commands and configuration to compose the available legacy Middleware, Network and Node layer capabilities as well as their innovative SPD functionalities.

Given an application scenario, it is possible to identify a legacy Middleware layer (e.g., CORBA, DCOM, etc.), composed by several Legacy Middleware Capabilities (e.g., remote procedure call, messaging queues, etc.). To be pSHIELD compliant, a Middleware layer must have a pSHIELD Middleware adapter, providing a mandatory set of Core SPD Services (i.e., discovery, composition, orchestration) and an optional set of innovative SPD functionalities (e.g., identification, authentication, auditing, accounting, anti-tampering, etc.).

3.4.1 SPD Driven Semantics

In order to cope with the intrinsic Embedded Systems complexity and heterogeneity the proposed solution is to design and implement a proper abstract, comprehensive semantic model able to provide a homogeneous representation of heterogeneous SPD-related parameters and conceptual models. So the idea is to abstract from Embedded Systems peculiarities and focus only on their abstract SPD semantic. In other words, the aim is to allow that heterogeneous SPD functionalities are seen by the Overlay as if they were a Seamless homogeneous pool.

To this purpose the technology-dependent SPD functionalities are abstracted in order to achieve a technological-independent framework; this task will be performed by extensively using semantic modelling techniques. As a consequence a uniform semantically-enriched ontological representation of carefully selected SPD functionalities can be achieved, which is fundamental in order to flexibly handle and control the considered Embedded System.

Semantic models in pSHIELD lean on the concept of ontology and ultimately shall enable interoperability at different levels in the conceptual framework of pSHIELD.

Given that the SPD domain in Embedded Systems shall be captured by a number of ontology, automatic reasoning is enabled in order to support several features of the pSHIELD framework.

Broadly speaking, a semantic engine (reasoner) shall enable interoperability within Middleware Layer and rule based discovery and composition within Overlay Agents, in such a way to provide the following essential enabling mechanisms:

- Semantic reasoning based on ontology models may carry out a reconciliation of heterogeneous formats of parameters exchanged between different layers (also suitable for interaction with legacy agents)
- The semantic characterization of the behavioural aspect of components makes it suitable for an agent to determine “what the service does”

The semantic characterization of the composition of functionalities and of the relations among them makes it suitable for an agent to reason about SPD metrics of the current configuration and – if needed - to carry out reconfigurations of the system at run-time, by means of rule-based combination / composition of components and SPD technologies, in order to achieve the new intended values for SPD metrics.

3.4.2 Core SPD Services

The *Core SPD Services* are a set of mandatory basic SPD functionalities provided by a pSHIELD Middleware Adapter with the aim to mediate the information exchange between the Overlay vertical layer and the three horizontal layers.

In particular, the Core SPD Services are in charge of:

- discovering and keeping updated the available SPD resources, services, functionalities, data and contextual information coming from the Node, Network and Middleware layer
- translating technology-dependent heterogeneous information into technology-independent metadata
- filtering, semantically enriching and aggregating mentioned metadata which will be eventually stored in the Semantic Knowledge Representation database of the pSHIELD Overlay
- enforcing the Overlay decisions all over the three horizontal layers, i.e. from the Middleware layer down to the Node layer, passing through the Network layer

The 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 an 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:

- secure service discovery
- service composition
- service orchestration

Secure service discovery allows any pSHIELD Middleware Adapter to discover in a secure manner the available SPD functionalities and services over heterogeneous environment, networks and technologies that are achievable by the pSHIELD Embedded System Device (pS-ESD) where it is running. Indeed the pSHIELD secure service discovery uses a variety of discovery protocols (such as SLP[1], SSDP[2], NDP[3], DNS[4], SDP[5], UDDI[6]) 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 an 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 be 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.

When the SPD services have been discovered and a feasible service composition has been identified, those services must be deployed, executed and continuously monitored. This is part of the **service orchestration** 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.

[1] IETF Service Location Protocol V2 - http://www.ietf.org/rfc/rfc2608.txt

[2] UPnP Simple Service Discovery Protocol - http://upnp.org/sdcps-and-certification/standards/

[3] IETF Neighbour Discovery Protocol - http://tools.ietf.org/html/rfc4861

[4] IETF Domain Name Specification - http://www.ietf.org/rfc/rfc1035.txt

[5] Bluetooth Service Discovery Protocol

[6] OASIS Universal Description Discovery and Integration - http://www.uddi.org/pubs/uddi_v3.htm

Secure service discovery, service composition and service orchestration operate at pSHIELD Middleware Layer and have access to the information coming from the Middleware, Network and Node layers as well as from the Overlay. These core SPD functionalities can take advantage from the information provided by the running services to "sense" the context or the situation in which the system is operating. Such a capability allows introducing an additional pSHIELD Middleware Adapter functionality that is *context awareness*. The context awareness is a pervasive functionality that is embedded in the discovery, composition and orchestration processes, thus it does not represent an individual core SPD functionality but an additional characteristic of the pSHIELD Middlware Adapter functionalities. In pSHIELD we introduce the context awareness of the pSHIELD Middleware Adapter core SPD services with a semantic approach, extending the service description model with context aware requirements[7].

3.4.3 Policy-based Management

A typical Policy Based Management (PBM) architecture is defined by the IETF policy framework [21]. The architecture constitutes several points and elements, i.e. Policy Management Tool (including the required tools and a policy repository), Policy Decision Point and a Policy Enforcement Point. The following figure presents an illustration of the main points in a typical PBM.

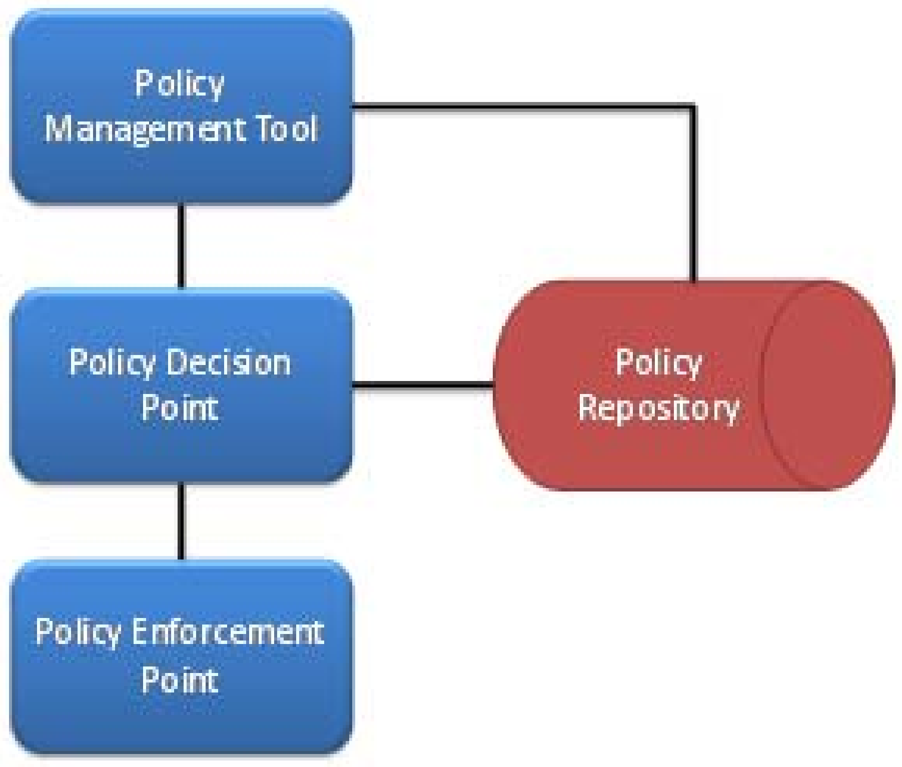


Figure 12 – Typical IETF PBM Architecture.

[7] V. Suraci, S. Mignanti, "Context-aware Semantic Service Discovery", 16th IST Mobile & Wireless Communications Summit Budapest, Hungary 1-7 July 2007. Proceedings. #273.

3.4.3.1 Policy Management Tool

Different terminologies are used to refer to PMT such as Policy Administration Point (PAP) for instance. PMT is mainly used by the administrator(s) in order to specify business-level (high-level) abstractions that constitute polices. A number of elements are needed at this point typically [21][22]:

1) A user interface that could be graphical with command-line support. Used as a policy editor with simple validation
2) A resource discovery element that determines the network topology, its capabilities, constituents, users and running applications
3) A policy translation (or transformation) element that transforms high-level policies into a lower-level constituents-specific policies. It also ensures policies' consistency, correctness and distribution feasibility through a validation process
4) A policy distributer/storage-retrieval element; as the name suggests, it interacts with the policy repository (explained onward) to store low-level policies and allow for their retrieval

An example of policy translation as described in [21] would be; assume a high-level policy segment that defines *"Premium Traffic between Point A and Point B"*. This can be translated into a low-level policy rule: *"source = 10.24.195.x, dest = 10.101.227.x, any protocol, perform Premium Service action".* Indeed, a validation check can only be carried out in an offline manner here where the syntactic and semantic integrity of each policy must be preserved. Some semantic validation checks are defined by [22] as in:

1) Bound checks: ensure that a given attribute value is within a predefined range
2) Relation checks: ensure that any two values assigned to interrelated policy parameters are satisfactory to their relationship
3) Consistency checks: ensure a conflict-free set of policies
4) Dominance checks: ensure that all specified policies are reachable and are to be active at some point during the system's lifetime
5) Feasibility checks: these are domain dependent checks that need to ensure that the underlying environment can support the specified policies

Moreover, while consistency checks need to ensure that conflicts among policy rules are avoided and given that this is an offline stage, other checks should be carried out at runtime to avoid potentially triggered conflicts.

3.4.3.1.1 Policy Repository

A policy repository is mainly concerned with managing translated policies, e.g., Directory Server, Database. It should allow for the storage, search and retrieval of policies and interface with other elements using for instance, a Lightweight Directory Access Protocol (LDAP) protocol.

3.4.3.1.2 Policy Decision Point

PDP is mainly a set of modules that are capable of examining applicable policies and consequently determine the decisions required for the system to comply with that policy. PDP is responsible for communicating policy-inferred decisions/actions to the Policy Enforcement Point (PEP) that could reside on several physical devices. That channel of communication is governed by a protocol such as SNMP. PDP also needs to interact, (i.e., fetch policies) with the policy repository using a protocol such as LDAP.

3.4.3.1.3 Policy Enforcement Point

PEP is the final point in a typical IETF policy framework architecture. PEPs act as logical entities that interface between systems' devices/resources such as sensors, where they are likely to reside, and the PDP by processing exchanged requests and responses. As the name suggests, PEP is responsible for enforcing actions communicated from the PDP at the device-level. Those actions reflect the policy or policies to be deployed at the local level.

3.5 Overlay Layer Definitions

Overlay: is a pSHIELD-specific vertical layer which is technology-independent and interoperates with the three pSHIELD horizontal layers (node, network and middleware) aiming at inter-layer SPD optimization and composability of heterogeneous SPD technologies.

In particular the Overlay is in charge of:

- elaborating, according to specific policies, the SPD related information coming from the horizontal layers
- taking consistent SPD related decisions concerning which SPD components have to be composed and the related configuration and composition rules
- enforcing the taken decisions back into the selected SPD components of the three horizontal layers

The Overlay takes its SPD composition decisions on the basis of a very rich information, consisting of dynamic, semantically enriched, multi-layer, aggregated SPD-related metadata expressed using a common, formal, technology-independent language.

The Overlay consists of a set of SPD Security Agents, each one controlling a given pSHIELD subsystem. The expandability of such a framework is obtained by enabling communication between SPD Security Agents controlling different sub-systems.

Security Agent (SA): a security agent is an entity, hardware or software, that performs Overlay functionalities. Usually there is one security agent per network, so that each agent is in charge of assuring SPD in its segment and, if necessary, exchange information with the neighbouring security agents if this is required to satisfy the application needs.

Semantic Information: the semantic information is a representation of hardware or software components that constitute the pSHIELD system and is modelled by means of ontologies.

4 SPD Considerations

4.1 Fundamental Concepts

4.1.1 Security and Privacy

4.1.1.1 Concepts

4.1.1.1.1 Cryptographic primitives

In the context of systems and network security, the basic functional security primitives that have been proposed and implemented are the various cryptographic algorithms that are to be used for data and communications encryption and decryption as well as data integrity. The three basic classes of cryptographic primitives are symmetric ciphers, asymmetric ciphers and hashing algorithms.

Symmetric ciphers perform the encryption and decryption operations using the same key that is to be shared among the sender and receiver of the message in hand. They are used mainly to ensure data confidentiality and are divided in two basic classes; block ciphers and stream ciphers. Block ciphers such as AES, DES, 3DES, perform the cryptographic operations on similar sized blocks of data, while stream ciphers such as RC4 operate on a bit by bit basis. On both cases, the cryptographic operations consist of a sequence of mathematical computations such as permutations and substitutions performed on the original or encrypted data.

Asymmetric ciphers, or public-key algorithms, such as RSA, Diffie-Hellman, use different keys for encryption and decryption. Those keys are called public and private keys. They use computationally intensive mathematical functions such as modular exponentiation and are used primarily for generating and verifying digital signatures and certificates as well as exchanging the symmetric cipher keys. Due to their computational complexity they are not usually used for encryption and decryption of the actual messages, but as a means to providing a secure channel for symmetric key exchange.

Hashing algorithms such as MD5 and SHA provide ways of mapping messages (with or without a key) into a fixed-length value, thereby providing “signatures” for messages. Thus, integrity checks can be performed on communicated messages by (a) having the sender “securely sending” the actual hash value of a message along with the message itself, (b) allowing the receiver to compute the hash value of the received message, and (c) comparing the two signatures to verify message integrity.

4.1.1.1.2 Security Mechanisms

Security solutions to meet the various security requirements typically rely on the aforementioned cryptographic primitives or on security mechanisms that use a combination of these primitives in a specific manner (e.g., security protocols). Various security technologies and mechanisms have been designed around these cryptographic algorithms in order to provide specific security services.

Security protocols provide ways of ensuring secure communication channels to and from the embedded system. IPSec [IPSec] and SSL [SSL] are popular examples of security protocols, widely used for Virtual Private Networks (VPNs) and secure web transactions, respectively.

Digital certificates provide ways of associating identity with an entity, while biometric technologies [Reid 2003] such as fingerprint recognition and voice recognition aid in end-user authentication. Digital signatures, which function as the electronic equivalent of handwritten signatures, can be used to authenticate the source of data as well as verify its identity.

Digital Rights Management (DRM) protocols, such as OpenIPMP [OpenIPMP], MPEG [MPEG], ISMA [ISMA], and MOSES [MOSES], provide secure frameworks for protecting application content against unauthorized use.

Secure storage and secure execution require that the architecture of the system be tailored for security considerations. Simple examples include the use of hardware to monitor bus transactions and block illegal accesses to protected areas in the memory, authentication of firmware that executes on the system, application isolation to preserve the privacy and integrity of code and data associated with a given application or process, HW/SW techniques to preserve the privacy and integrity of data throughout, the memory hierarchy, execution of encrypted code and so on.

4.1.1.1.3 Design Challenges

Designers of a large and increasing number of embedded systems need to support various security solutions in order to deal with one or more of the security requirements described earlier. These requirements present significant bottlenecks during the embedded system design process.

Processing Gap: Existing embedded system architectures are not capable of keeping up with the computational demands of security processing, due to increasing data rates and complexity of security protocols. These shortcomings are most felt in systems that need to process very high data rates or a large number of transactions (e.g., network routers, firewalls, and web servers), and in systems with modest processing and memory resources (e.g., PDAs, wireless handsets, and smartcards). In this paper, we will examine the two sides of the processing gap issue (requirements and availability) and study various solutions proposed to address this mismatch.

Battery Gap: The energy consumption overheads of supporting security on battery-constrained embedded systems are very high. Slow growth rates in battery capacities (5–8% per year) are easily outpaced by the increasing energy requirements of security processing, leading to a battery gap. Various studies [Carman et al. 2000; Perrig et al. 2002; Potlapally et al. 2003] show that the widening battery gap would require designers to make energy-aware design choices (such as optimized security protocols, custom security hardware, and so on) for security.

Flexibility: An embedded system is often required to execute multiple and diverse security protocols and standards in order to support (i) multiple security objectives (e.g., secure communications, DRM, and so on), (ii) interoperability in different environments (e.g., a handset that needs to work in both 3G cellular and wireless LAN environments) and (iii) security processing in different layers of the network protocol stack (e.g., a wireless LAN enabled PDA that needs to connect to a virtual private network, and support secure web browsing may need to execute WEP, IPSec, and SSL). Furthermore, with security protocols being constantly targeted by hackers, it is not surprising that they keep continuously evolving. It is, therefore, desirable to allow the security architecture to be flexible (programmable) enough to adapt easily to changing requirements. However, flexibility may also make it more difficult to gain assurance of a design's security.

Tamper Resistance: Attacks due to malicious software such as viruses and trojan horses are the most common threats to any embedded system that is capable of executing downloaded applications [Howard and LeBlanc 2002; Hoglund and McGraw 2004; Ravi et al. 2004]. These attacks can exploit vulnerabilities in the operating system (OS) or application software, procure access to system internals and disrupt its normal functioning. Because these attacks manipulate sensitive data or processes (integrity attacks), disclose confidential information (privacy attacks) and/or deny access to system resources (availability attacks), it is necessary to develop and deploy various HW/SW countermeasures against these attacks. In many embedded systems such as smartcards, new and sophisticated attack techniques, such as bus probing, timing analysis, fault induction, power analysis, electromagnetic analysis and so on, have been demonstrated to be successful in easily breaking their security [Ravi et al. 2004; Anderson and Kuhn 1996, 1997; Kommerling and Kuhn 1999; Rankl and Effing; Hess et al. 2000; Quisquater and Samyde

2002; Kelsey et al. 1998]. Tamper resistance measures must, therefore, secure the system implementation when it is subject to various physical and side-channel attacks. Later in this paper (see Section 6), we will discuss some examples of embedded system attacks and related countermeasures.

Assurance Gap: It is well known that truly reliable systems are much more difficult to build than those that merely work most of the time. Reliable systems must be able to handle the wide range of situations that may occur by chance. Secure systems face an even greater challenge: they must continue to operate reliably despite attacks from intelligent adversaries who intentionally seek out undesirable failure modes. As systems become more complicated, there are inevitably more possible failure modes that need to be addressed. Increases in embedded system complexity are making it more and more difficult for embedded system designers to be confident that they have not overlooked a serious weakness.

Cost: One of the fundamental factors that influence the security architecture of an embedded system is cost. To understand the implications of a security related design choice on the overall system cost, consider the decision of incorporating physical security mechanisms in a single-chip cryptographic module. The Federal Information Processing Standard (FIPS 140-2) [FIPS] specifies four increasing levels of physical (as well as other) security requirements that can be satisfied by a secure system. Security Level 1 requires minimum physical protection, Level 2 requires the addition of tamper-evident mechanisms such as a seal or enclosure, while Level 3 specifies stronger detection and response mechanisms. Finally, Level 4 mandates environmental failure protection and testing (EFP and EFT), as well as highly rigorous design processes. Thus, we can choose to provide increasing levels of security using increasingly advanced measures, albeit at higher system costs, design effort, and design time. It is the designer’s responsibility to balance the security requirements of an embedded system against the cost of implementing the corresponding security measures.

4.1.2 Dependability

The protection and survival of networked systems and services that are running on those systems is the main concern of an SPD driven architecture. In different circumstances, different properties of such services are the ones that we focus on sustaining. These are, per example, the average response time of the service, the capability to prevent intrusions, the ability to produce the required output and the ability to survive a catastrophic failure.

Dependability sums up those concerns under a common framework. We can identify three main parts to the concept of dependability.

- The threats
- The attributes
- The means

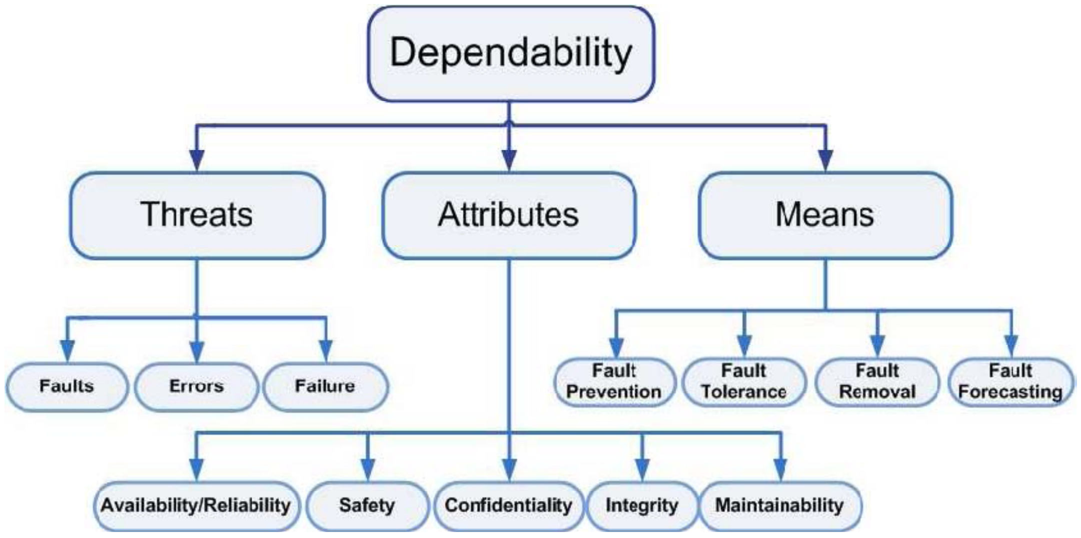


Figure 13– Dependability concept taxonomy.

The five fundamental properties of a computing system are

- Functionality
- Usability
- Performance
- Cost
- Dependability

Dependability is the ability of the system to deliver a service in a justifiable trusted way. The function of the system is the intended output of the system and is described in the functional specifications of the system. Correct service is delivered when the system is providing the intended function as per specification. System failure is the incorrect service provisioning. Dependability can also be described as the ability of a system to avoid failures that are more frequent, more severe and last longer than the user's expectations.

4.1.2.1 The threats

A system failure can occur when either the system does not comply with the functional specification or the specification does not describe adequately its function. An error is the part of the system state that can introduce a failure. The actual failure occurs when the error reaches the system's interface. The fault is the cause of an error. The ways in which a system can fail are its failure modes and can be ranked according to severities.

A system is the whole of interacting components; therefore a system state is the set of its component states. A fault causes an error on one or more system components. A system failure occurs only when those errors reach the service interface of the system.

We identify three major fault classes

- Design faults
- Physical faults
- Interaction faults

The semantics of the terms fault, error and failure reflect the current usage

- Fault prevention, tolerance, diagnosis
- Error detection, correction
- Failure rate, failure mode

4.1.2.2 The attributes

Dependability is a concept composed by the following basic attributes

- Availability
- Reliability
- Safety
- Confidentiality
- Integrity
- Maintainability

The description of the required goals of those attributes in terms of frequency, severity and duration of failure modes for a specific set of failure modes in a specific environment, is the dependability requirement of the system.

Depending on the indented application of the system, those attributes contribute in different weights to the dependability requirement. Availability is always a prerequisite, but reliability, safety and confidentiality may be required to a limited degree in certain applications.

Integrity is a prerequisite for availability, reliability and safety but not always for confidentiality. For example, attacks via passive listening can lead to loss of confidentiality without threatening integrity.

Security, although not included as a single attribute of dependability, can be described as the combination of confidentiality, integrity and availability.

There are also some secondary attributes, especially relevant to security, which can be defined when we distinguish amongst various types of information

- Accountability: availability and integrity of the identity of the person that performs an operation
- Authenticity: integrity of the source and content of a message and other attributes of the message such as time of emission
- Non-repudiation: availability and integrity of the identity of the sender and receiver of a message

The different weights that are put on those attributes directly affect the means that are to be used in order to make the resulting system dependable. Most of the times, those attributes are also conflicting with each other and several design trade-offs are required.

4.1.2.3 The means

The four main techniques utilized for the development of a dependable computing system are the following

- Fault prevention: It is attained by employing quality control during the design and implementation phases. For software, these include structured programming and modularization and for hardware, rigorous design rules. Physical faults are prevented through shielding, radiation hardening etc. Interaction faults are prevented through training and rigorous procedures. Malicious faults are prevented through the use of firewalls, intrusion detection systems and similar defences
- Fault tolerance: Is the notion of delivering the correct service even during the presence of active faults. This is achieved by error detection and system recovery in the forms of rollback,

compensation or roll forward. Fault tolerance is not restricted to accidental faults. Malicious faults are also the target of the error detection mechanisms

- Fault removal: This is preformed during both development and operational phases. During the development life cycle, fault removal is consisting of three steps; verification, diagnosis, correction. During the normal operational phase, fault removal is performed via corrective and pre-emptive maintenance. Pre-emptive maintenance aims on removing faults before they cause errors during operation

Fault forecasting: Is the outcome of the evaluation of the system behaviour with respect to fault occurrence. The main metric used in this process is failure intensity. The alteration of correct-incorrect service delivery is quantified to define reliability, availability and maintainability as measures of dependability.

4.2 Embedded Systems

4.2.1 Introduction

In addition to the typical requirements for responsiveness, reliability, availability, robustness and extensibility, many conventional embedded systems and applications have significant security requirements. However, security is a resource-demanding function that needs special attention in embedded computing. Furthermore, the wide deployment of small devices which are used in critical applications has triggered the development of new, strong attacks that exploit more systemic characteristics, in contrast to traditional attacks that focused on algorithmic characteristics, due to the inability of attackers to experiment with the physical devices used in secure applications. Thus, design of secure embedded systems requires special attention.

4.2.2 Design of Secure Embedded Systems

Secure embedded systems must provide basic security properties, such as data integrity, as well as mechanisms and support for more complex security functions, such as authentication and confidentiality. Furthermore, they have to support the security requirements of applications, which are implemented, in turn, using the security mechanisms offered by the system.

4.2.2.1 System Design Issues

Design of secure embedded systems needs to address several issues and parameters ranging from the employed hardware technology to software development methodologies. Although several techniques used in general-purpose systems can be effectively used in embedded system development as well, there are specific design issues that need to be addressed separately, because they are unique or weaker in embedded systems, due to the high volume of available low cost systems that can be used for development of attacks by malicious users. The major of these design issues are tamper-resistance properties, memory protection, Intellectual Property (IP) protection, management of processing power, communication security and embedded software design.

Modern secure embedded systems must be able to operate in various environmental conditions, without loss of performance and deviation from their primary goals. In many cases they must survive various physical attacks and have tamper resistance mechanisms. Tamper resistance is the property that enables systems to prevent the distortion of physical parts. Additionally to tamper resistance mechanisms, there exist tamper evidence mechanisms, which allow users or technical stuff to identify tampering attacks and take countermeasures.

IP protection of manufacturers is an important issue addressed in secure embedded systems. Complicated systems tend to be partitioned in smaller independent modules leading to module reusability and cost reduction. These modules include IP of the manufacturers, which needs to be protected from third–party users, who might claim and use these modules. The illegal users of an IP block do not necessarily need to have full, detailed knowledge of the IP component, since IP blocks are independent modules which can very easily incorporated and integrated with the rest of the system components

Implementation of security techniques for tamper resistance, tamper prevention and IP protection may require additional processing power, which is limited in embedded systems. The “processing gap” between the computational requirements of security and the available processing power of embedded processors requires special consideration. Available technologies include use of cryptographic co-processors and accelerators, embedded security processors and programmable security protocols. One other approach is to develop enhancements in the Instruction Set Architecture (ISA) of embedded processors, in order to efficiently calculate various cryptographic primitives, such us permutations, bit rotations, fast substitutions and modular arithmetic or even build dedicated cryptographic embedded co-processors with their own ISA.

Even if the “processing gap” is bridged and security functions are provided, embedded systems are required to support secure communications as well, considering that, often, embedded applications are implemented in a distributed environment where communicating systems may exchange (possibly) sensitive data over an untrusted network –wired, wireless or mobile- like Internet, a Virtual Private Network, the Public Telephone network, etc. In order to fulfil the basic security requirements for secure communications, embedded systems must be able to use strong cryptographic algorithms and to support various protocols. One of the fundamental requirements regarding secure protocols is interoperability, leading to the requirement for system flexibility and adaptability. Since an embedded system can operate in several environments, e.g. a mobile phone may provide 3G cellular services or connect to a wireless LAN, it is necessary for the system to operate securely in all environments without loss of performance. Furthermore, as security protocols are developed for various layers of the OSI reference model, embedded systems must be adaptable to different security requirements at each layer of the architecture.

Embedded software, such as the operating system or application-specific code, constitutes a crucial factor in secure embedded system design. There are three basic factors that make embedded software development a challenging area of security: (a) complexity of the system, (b) system extensibility and (c) connectivity. Embedded systems serve critical, complex, hard to implement applications with many parameters that need to be considered, which, in turn, leads to “buggy” and vulnerable software. Furthermore, the required extensibility of conventional embedded systems makes the exploitation of vulnerabilities relatively easy. Finally, as modern embedded systems are designed with network connectivity, the higher the connectivity degree of the system, the higher the risk for a software breach to expand as time goes by. Many attacks can be implemented by malicious users that exploit software glitches and lead to system unavailability, which can have a disastrous impact, e.g. a Denial-of-Service attack on a military embedded system.

4.2.2.2 Application Design Issues

Embedded system applications present significant challenges to system designers, in order to achieve efficient and secure systems. A key issue in secure embedded design is user identification and access control. User identification includes the necessary mechanisms that guarantee that only legitimate users have access to system resources and can also verify, whenever requested, the identity of the user who has access to the system. A solution to this problem may come from an emerging new technology for user identification which is based on biometric recognition, for both user identification and verification. Biometrics are based on pattern recognition in acquired biological data taken from a user who wants to gain access to a system, i.e. palm prints, finger prints, iris scan, etc., and comparing them with the data that have been stored in databases identifying the legitimate users of the system. A secure smart card which uses biometrics capabilities is less vulnerable to attacks when compared to software based solutions and that the combination of smartcard and fingerprint recognition is much more robust than PIN-based identification.

As mentioned previously, an embedded system must store information that enables it to identify and validate users that have access to the system. But, how does an embedded system store this information? Embedded systems use several types of memory to store different types of data: (i) ROM EPROM to store programming data used to serve generic applications, (ii) RAM to store temporary data, and (iii) EEPROM and FLASH memories to store mobile downloadable code. In an embedded device such as a PDA or a mobile phone several pieces of sensitive information like PINs, credit card numbers, personal data, keys and certificates for authorization purposes, may be permanently stored in secondary storage media. The requirement to protect this information as well as the rapid growth of communications capabilities of embedded devices, which make embedded systems vulnerable to network attacks as well, lead to increasing demands for secure storage space. The use of hard cryptographic algorithms to ensure data integrity and confidentiality is not feasible in most embedded systems, mainly due to their limited computational resources.

Significant attention has to be paid to protect against possible attacks through malicious downloadable software, like viruses, Trojans, logic bombs, etc. The wide deployment of distributed embedded systems and the Internet have resulted to the requirement for ability of portable embedded systems, e.g. mobile phones and PDAs, to download and execute various software applications. This ability may be new to the world of portable, highly constrained embedded systems, but it is not new in the world of general–purpose systems, which have had the ability to download and execute files from the Internet or from other network resources for a long time. One major problem in this service is that users cannot be certain about the content of the software that is downloaded and executed on their system(s), about who the creator is and what its origin is. An additional important consideration is the robustness of the downloadable code: once the mobile code is considered secure, downloaded and executed, it must not affect preinstalled system software.

4.2.3 Cryptography and Embedded Systems

Secure embedded systems should support the basic security functions for (a) confidentiality, (b) integrity, and (c) authentication. Cryptography provides a mechanism that ensures that the previous three requirements are met. However, implementation of cryptography in embedded systems can be a challenging task. The requirement of high performance has to be achieved in a resource-limited environment; this task is even more challenging when low power constraints exist. Performance usually dictates an increased cost, which is not always desirable or possible. Cryptography can protect digital assets provided that the secret keys of the algorithms are stored and accessed in a secure manner. For this, the use of specialized hardware devices to store the secret keys and to implement cryptographic algorithms is preferred over the use of general-purpose computers. However, this also increases the implementation cost and results in reduced flexibility. On the other hand, flexibility is required, because modern cryptographic protocols do not rely on a specific cryptographic algorithm but rather allow use of a wide range of algorithms for increased security and adaptability to advances on cryptanalysis. For example, both the SSL and IPSec network protocols support numerous cryptographic algorithms to perform the same function, such as encryption. The protocol enables negotiation of the algorithms to be used, in order to ensure that both parties use the desirable level of protection dictated by their security policies.

Apart from the performance issues, a correct cryptographic implementation requires expertise that is not always available or affordable during the lifecycle of a system. Insecure implementations of theoretically secure algorithms have made their way to headline news quite often in the past. The cryptographic community has focused on proving the theoretical security of various cryptographic algorithms and has paid little attention to actual implementations on specific hardware platforms. In fact, many algorithms are designed with portability in mind and efficient implementation on a specific platform meeting specific requirements can be quite tricky. This communication gap between vendors and cryptographers intensifies in the case of embedded systems, which can have many design choices and constraints that are not easily comprehensible.

In the late 1990s, side-channel attacks were introduced. Side-channels attacks are a method of cryptanalysis that focuses on the implementation characteristics of a cryptographic algorithm in order to derive its secret keys. This advancement bridged the gap between embedded systems, a common target of such attacks, and cryptographers. Vendors became aware and concerned by this new form of attacks, while cryptographers focused on the specifics of the implementations, in order to advance their cryptanalysis techniques.

4.2.3.1 Physical Security

Secrecy is always a desirable property. In the case of cryptographic algorithms, the secret keys of the algorithm must be stored, accessed, used and destroyed in a secure manner, in order to provide the required security functions. This statement is often overlooked and design or implementation flaws result to insecure cryptographic implementations. It is well-known that general purpose computing systems and operating systems cannot provide enough protection mechanisms for cryptographic keys.
Embedded systems are commonly used for implementing security functions. Since they are complete systems, they can perform the necessary cryptographic operations in a sealed and controlled environment. Tamper resistance refers to the ability of a system to resist to tampering attacks, i.e., attempts to bypass its attack prevention mechanisms. Smart cards are a well-known example of tamper resistant embedded systems that are used for financial transactions and subscription-based service provision.

In many cases, embedded systems used for security-critical operations do not implement any tamper resistance mechanisms. Rather, a thin layer of obscurity is preferred, both for simplicity and performance issues. However, as users become more interested in bypassing the security mechanisms of the system, the thin layer of obscurity is easily broken and the cryptographic keys are publicly exposed.

Finally, an often neglected issue is a lifecycle-wide management of cryptographic systems. While a device may be withdrawn from operation, the data it has stored or processed over time may still need to be protected. The security of keys that relies on the fact that only authorized personnel has access to the system may not be sufficient for the recycled device.

4.2.3.2 Side-channel cryptanalysis

Until the middle 1990s, academic research on cryptography focused on the mathematical properties of the cryptographic algorithms. Paul Kocher was the first to present cryptanalysis attacks on implementations of cryptographic algorithms, which were based on the implementation properties of a system. Kocher observed that a cryptographic implementation of the RSA algorithm required varying amounts of time to encrypt a block of data depending on the secret key used. Careful analysis of the timing differences, allowed him to derive the secret key and he extended this method to other algorithms as well. This result came as a surprise, since the RSA algorithm has withstood years of mathematical cryptanalysis and was considered secure. These findings revealed a new class of attacks on cryptographic algorithms. The term side-channel attacks (SCA), has been widely used to refer to this type of cryptanalysis, while the terms fault-based cryptanalysis, implementation cryptanalysis, active/passive hardware attacks, leakage attacks and others have been used also. Cryptographic algorithms acquired a new security dimension, that of their exact implementation. Cryptographers had previously focused on understanding the underlying mathematical problems and prove or conjecture for the security of a cryptographic algorithm based on the abstract mathematical symbols. Now, in spite of the hard underlying mathematical problems to be solved, an implementation may be vulnerable and allow the extraction of secret keys or other sensitive material. Implementation vulnerabilities are of course not a new security concept. The new concept of SCA is that even cryptographic algorithms that are otherwise considered secure can be also vulnerable to such faults.

This observation is of significant importance, since cryptography is widely used as a major building block for security; if cryptographic algorithms can be driven insecure, the whole construction collapses.

In the following, we present the classes of side-channel attacks and countermeasures that have been developed. Embedded system vendors must study the attacks carefully, evaluate the associated risks for their environment, and ensure that appropriate countermeasures are implemented in their systems; furthermore, they must be prepared to adapt promptly to new techniques for deriving secrets from their systems.

4.2.3.3 Side channel implementations

A side channel is any physical channel that can carry information from the operation of a device while implementing a cryptographic operation; such channels are not captured by the existing abstract mathematical models. The definition is quite broad and the inventiveness of attackers is noticeable. Timing differences, power consumption, electromagnetic emissions, acoustic noise and faults have been currently exploited for leaking information out of cryptographic systems. The channel realization can be categorized in three broad classes: physical or probing attacks, fault-induction or glitch attacks and emission attacks, like TEMPEST. We shortly review the first two classes:

The side channels may seem unavoidable and a frightening threat. However, it should be strongly emphasized that in most cases, reported attacks, both theoretical and practical, rely for their success on the detailed knowledge of the platform under attack and the specific implementation of the cryptographic algorithm.

4.2.3.3.1 Fault induction techniques

Devices are always susceptible to erroneous computations or other kinds of faults for several reasons. Faulty computations are a known issue from space systems, because, in deep space, devices are exposed to radiation which can cause temporary or permanent bit flips, gate destruction, or other problems. Incomplete testing during manufacturing may allow imperfect designs from reaching the market or in the case of device operation in conditions out of their specifications. Careful manipulation of the power supply or the clock oscillator can also cause glitches in code execution by tricking the processor for example to execute unknown instructions or bypass a control statement. Some researchers have questioned the feasibility of fault-injection attacks on real systems. While fault injection may seem as an approach that requires expensive and specialized equipment, there have been reports that fault injection can be achieved with low cost and readily available equipment.

The combined time-space isolation problem is of significant importance in fault-induction attacks. The space isolation problem refers to isolation of the appropriate space (area) of the chip in which to introduce the fault. The space isolation problem has four parameters:

- *Macroscopic*: the part of the chip where the fault can be injected. Possible answers can be one or more of the following: main memory, address bus, system bus, register file
- *Bandwidth*: the number of bits that can be affected. It may be possible to change just one bit or multiple bits at once. The exact number of changed bits can be controllable (e.g., one) or follow a random distribution
- *Granularity*: the area where can the error occur. The attacker may drive the fault injection position at a bit level or a wider area, such as a byte or a multi-byte area. The fault injected area can be covered by a single error or by multiple errors. How are these errors distributed with respect to the area? They may focus around the mark or evenly distributed
- *Lifetime*: the time duration of the fault. It may be a transient fault or a permanent fault. For example, a power glitch may cause a transient fault at a memory location, since the next time the location will be written, a new value will be correctly written. In contrast, a cell or gate destruction will result in a permanent error, since the output bit will be stuck at 0 or 1, independently of the input

The time isolation problem refers to the time at which a fault is injected. An attacker may be able to synchronize exactly with the clock of the chip or may introduce the error in a random fashion. This granularity is the only parameter of the time isolation problem. Clearly, the ability to inject a fault in a clock period granularity is desirable, but impractical in real world applications.

4.2.3.3.2 Passive side channels

Passive side channels are not a new concept in cryptography and security. The information available from the now partially declassified TEMPEST project reveals helpful insights in how electromagnetic emissions occur and can be used to reconstruct signals for surveillance purposes. The new concept in this area is the fact that such emissions can be also used to derive secret information from an otherwise secure device.

Researchers have been quite creative and have used many types of emissions or other physical interactions of the device with the environment it operates. A basic idea is the monitoring of execution time of a cryptographic algorithm and tries to identify the secret keys used. The key concept in this approach is that an implementation of an algorithm may contain branches and other conditional execution or the implementation may follow different execution paths. If these variances are based on the bit values of a secret key, then a statistical analysis can reveal the secret key bit by bit. Power consumption can be also correlated with key bits.

4.3 Applications

Nowadays railway is considered as one of the most important critical infrastructures due to its importance not only in mass transport systems but also its usability in material and goods transport. In pSHIELD railway infrastructures and operations had been chosen as the potential areas where the results of pSHIELD can contribute to ensure its secure, dependable and reliable operations. More specifically, pSHIELD focuses on hazardous material transport scenario by railways system. Within this specific scenario, the following applications are planned which focus at pSHIELD core features such as embedded systems, SPD considerations, sensor integration and composability.

Monitoring

In hazardous material transport, due to the nature of goods, continuous monitoring is crucial for safe and dependable operations of such transport scenario. For example, the following situations may threat safety of the goods:

- Some materials must not be collocated with inflammable substances
- The on-carriage temperature should not rise above a certain level
- The speed of the passing train should not be more than a certain limit as excess speed may cause higher vibration
- Only authorized personnel should get access to the carriage

To monitor these situations, the carriage is expected to be equipped with various types of sensors to collect e.g. temperature, pressure, vibration information. Besides, the carriage contains recoding camera and position sensors. All the information can be stored on-board and can be transmitted to the control centre of the railway systems on-demand. In order to facilitate remote communication, the carriage is equipped with multiple communication systems. Above all, as contingency, such carriage carries extra power source in case of un-availability of power from the grid.

Interoperable Railway Information System

In such hazardous material transport scenario, different stakeholders are involved. For safety of their infrastructures or goods, the stakeholders are willing to get access to the data being transmitted from the carriage. However, while sharing the monitored data among them privacy of some critical data needs to be ensured. The following scenario will further exemplify the privacy concerns:

DHL Norway receives a booking from a chemical plant to transport hazardous material from its industrial plant. DHL then rents a cargo from a railway operator and puts the cargo into its carriage. Here the following stakeholder are involved: a) the railway administration (operating the whole railway network), b) train operators (runs trains and freights), c) first party consumer (e.g. DHL Norway), and d) second party consumer (e.g. chemical plant).

Due to nature of the cargo each of these stakeholders may want to monitor cargo being transported. Now the question is to what extent the data will be delivered to the concerned parties preserving privacy of the freight and its owner information. For example, it is not necessary to inform the railway administration about the owner of the goods whereas the administration must know the content of the freight being transported.

This application goes beyond the on-carriage monitoring situation and extends towards preservation of privacy of various information, within the hazardous material transport scenario.

4.4 Preliminary Concept of pSHIELD Demonstrator Architecture

4.4.1 System architecture for monitoring

The pSHIELD scenario is addressing rail transportation of dangerous goods by a wagon. The pSHIELD demonstrator will exhibit the typical monitoring application such as monitoring of on-carriage temperature remotely. This section introduces the preliminary system architecture of such monitoring applications using the following figure where the prototypical railway wagon is called Intelligent Wagon (IW).

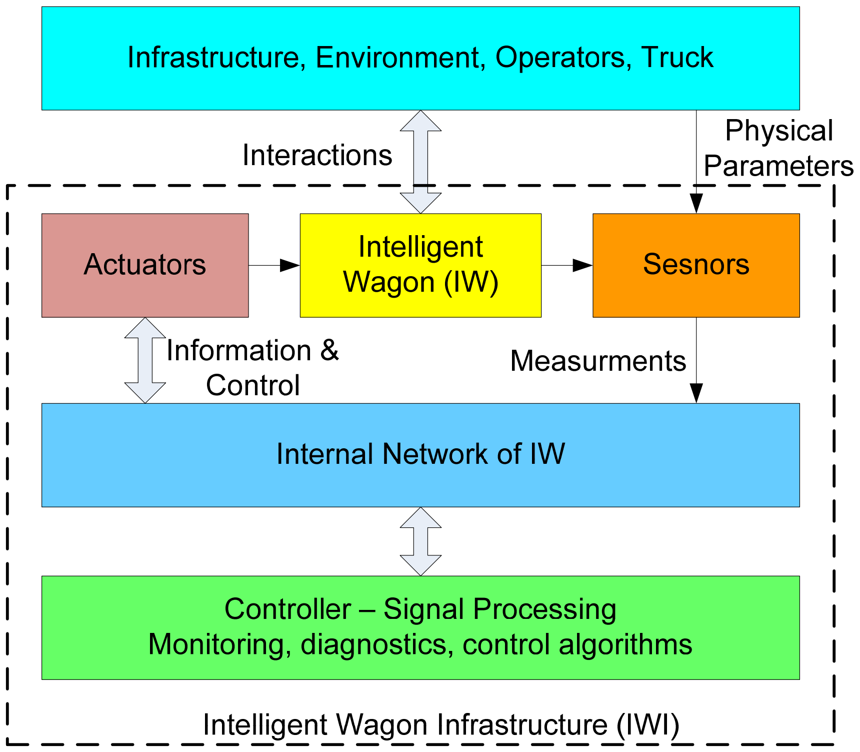

Infrastructure, Environment, Operators, Truck
Interactions
Physical Parameters
Actuators
Intelligent Wagon (IW)
Sesnors
Information & Control
Measurments
Internal Network of IW
Controller – Signal Processing
Monitoring, diagnostics, control algorithms
Intelligent Wagon Infrastructure (IWI)

Figure 14 – Intelligent Wagon Infrastructure.

The railway freight cars would be filled with sensors & intrusion detection devices, gateway node for sensors, control unit featuring embedded CPU and OS, GPS and communication interfaces such as GSM. The sensors will sense the physical environment and send the data to the control unit through internal network of IW. The control unit based on predefined logic will detect abnormal events. The control unit then initiates the transmission of warning message and event data to the railway control center. 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. Thus both natural and malicious faults can have an impact on system availability and indirectly on safety. The monitoring system of pSHIELD demonstrator is aimed to the detection of abnormal operating or environmental conditions on board of vehicles as well as threats of burglary. For these two cases, the basic working logic of the system is as follows:

- Case of abnormal environmental or operational conditions: whenever an abnormal event (e.g. very high temperature or out of range vibrations) is detected by sensor/sensors, its transmission unit is activated and data is received by gateway node. Central unit having CPU and OS validates data and - if the anomaly is confirmed - it activates GPS to achieve the current position and communication interfaces to send an appropriate warning message to the control center

Case of intrusions: whenever intrusion detection device detects an opening, central unit activates GPS to achieve the current position and communication interface to send an appropriate alarm message to the control centre.

4.4.2 System architecture for Integration and Interoperation for IRIS

One of the objectives of pSHIELD is to achieve Integration and Interoperation of heterogeneous services, systems and devices. The pSHIELD demonstrator will address the interoperability of information from Railway Infrastructure domain to third party service providers through an M2M platform provided by Telenor Objects, Norway. The demonstrator will make the information coming from the sensors at the Intelligent Wagon Infrastructure (IWI) accessible from anywhere at any time. However it is assumed that only authorized services providers can access them. pSHIELD demonstrator will address the integration of sensor data with the M2M platform. The figure below illustrates the concept of Integration and Interoperation of heterogeneous services, systems and devices in pSHIELD demonstrator.

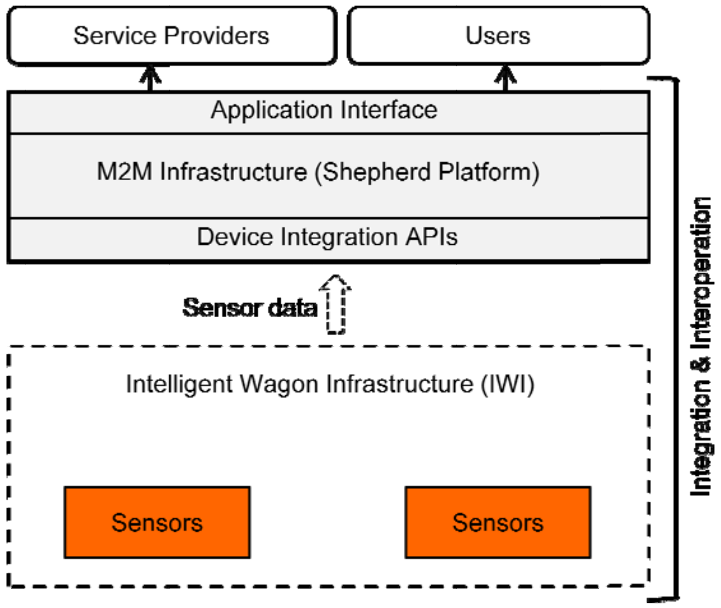

Service Providers
Users
Application Interface
M2M Infrastructure (Shepherd Platform)
Device Integration APIs
Sensor data
Intelligent Wagon Infrastructure (IWI)
Sensors
Sensors
Integration & Interoperation

Figure 15 – Integration & Interoperation concept of heterogeneous services.

Shepherd platform provided by Telenor Objects acts as M2M platform. Shepherd allows any pluggable objects (here micro and power nodes) to be connected to the platform through devices APIs and makes the sensor information securely and reliably available to the service providers or users through application interface.

4.5 System Architecture SPD functionalities

Following are the core SPD functionalities pSHIELD envisioned:

- **Identification:** Identification process ascertains the identity of an entity. For example, the process validates that the individual or process presenting the identity is indeed the owner of the identity. pSHIELD system includes the identification process of several of its components such as nodes, applications, processes and individuals. A simple identification process in the middleware layer can validate the authorized nodes of the system.

- **Authentication:** An entity usually makes claims about itself for identification towards a system. Then the system needs to verify the claims. A part of the entity's identity attribute is used to verify that the claims made by it about itself are true. Authentication is the process of identifying an individual who wants to access a system. Access is granted when the presented claims is equal to the information stored in the system. An entity can be a node, an application, process or an individual.
- **Security accounting and audit:** Accounting functionality tracks security events such as authentication and authorization failures. It mainly monitors the system from security point of views and keeps record of the events. Accounting includes the audit functionality. Security audit refers to systematic and measurable assessment of security of system or application. It includes mainly security vulnerability analysis of system. A system may generate audit reports using software. Security audit can also be manual.
- **Integrity:** The pSHIELD railway freight transportation scenario mainly aims at monitoring hazardous materials transported by trains in carriages equipped with a wireless sensor network, devices for intrusion detection and access control and a location-aware communication transceiver. Having a wireless monitoring system installed per train carriage increases the need for security. It is essential, given the hazardous nature of the transported materials, to eliminate any risk of harming human lives or the environment through malicious attacks. Cryptographic functionality will be adopted to counter these malicious attacks. Cryptography has been an established mean for many years to provide security and information protection against different forms of attacks. It is seen as the basis for the provision of different systems security, fundamentally by seeking to achieve a number of goals, that are: confidentiality, authenticity, data integrity and non-repudiation.

5 SPD Requirements

Architecture is one of the structural characteristics of pSHIELD. The process for the definition of a conceptual formalized architectural framework demands as inputs a set of requirements, which are of several types or simply express different system needs and priorities. We assign the term “System Requirements” in the collection concerning the overall design. This can be further decomposed in attributes independent of the application or reflecting the specific scenario needs. We aim at defining the SPD functionalities, which will be potentially possessed by a pSHIELD implementation, in order to serve its objective, being for example, the protection of asset and material in the railway transportation use case. These requirements pave the way for the design and development of pSHIELD Architecture, synthesized by the four pSHIELD layers of node, network, middleware and overlay.

Out of the broad list of D2.1, “System requirements and specifications”, we refer here to these requirements, which relate, to a bigger or lesser extent, with the impact of specific SPD desired attributes, metrics and functionalities, in the formulation of the system architecture. Before that, an epigrammatic reminder of the three components comprising SPD concept and their meaning in the technological framework in pSHIELD follows.

Security in the context of telecommunications and ESs is the resultant of three properties: confidentiality, integrity and availability. Their synthesis represents the discipline of protecting software and hardware against attacks conducted by unauthorized interceptors.

Apart from the apparent, colloquial meaning, **Privacy**, expanded respectively in communication theory, is the seclusion or selective revelation of wire, oral or electronic communication while in transmission. Usually privacy is a notion broader than security. In pSHIELD, in expectation of an improved specification of the term (probably through the application scenario use case), privacy can be a complementary and interrelated term to security and more specifically, confidentiality.

Even broader and multi-faceted is **Dependability,** a notion arriving in telecommunications from system engineering and encompassing the reliability and trustworthiness of a system or network. As mentioned in D2.1, in pSHIELD, Dependability embraces the meaning of availability, reliability, safety, integrity and maintainability.

The description of each requirement is accompanied by the code with which it can be found (itself or a set of similar phrased ones) in D2.1. Either these requirements are tightly connected to architectural characteristics or they imply features and functionalities dependent on the development of the architecture proposal.

5.1 System Architecture Security Requirements

Availability
The attribute concerns availability of information, node, network and system, for authorized users (0301, 01003, 01014).

Integrity
Of data (network layer), against unauthorized access, mechanisms based on hardware “hooks” and secure key installation, protection of the TPM, integrity at node layer (0302, 06004, 01002, 01004, 01012, 01015).

Central control unit
The system should have a central control unit and the respective monitoring applications should have reliable communication links between the peripheral nodes and this unit (20003, 06001).

Confidentiality aware information delivery
In the network layer, data confidentiality to protect and encrypt the transmitted information should be supported (06005, 20035).

Audit functionalities
pSHIELD system should guarantee audit functionalities (06006).

Encryption
pSHIELD system should guarantee encryption functionalities, TPM cryptographic protocols and improvement, cryptography on node layer, encryption algorithms on network layer (06007, 01008, 01011, 09002, 01031, 20036).

Non-repudiation
pSHIELD system should provide non-repudiation functionalities (06008).

Access control
pSHIELD system should establish access control functionalities among users, assets and operations on system and node level (06009, 01003, 01030).

Identification and authentication
The pSHIELD system should guarantee identification and authentication functionalities, including node and network layers (06010, 01031, 20038).

Tamper Resistance
There should be anti-tampering functionalities for physical attacks on nodes (06012).

Timestamps
The pSHIELD system should be able to provide reliable timestamps (06013).

Trusted channel
The pSHIELD system should provide trusted channel for SPD functionalities (06015).

Secure service discovery
Middleware should support secure service discovery (20029).

Management of security functionalities
There should be efficient management of the above listed security functionalities (06011).

Network security
On network layer, security protocols should be implemented, including the protection of IP or upper layers payload and protocols (20031, 20032).

Denial of service

Anti-replay protection should protect against denial of service attacks (20034).

IPSec
There should be applicability of IPSec at network layer (20039).

WS-Security
There should be applicability of Web Services security at Middleware layer (20045).

Security Agent
The formulation and function of Security Agent encapsulates pSHIELD platform and scope. It is a module thoroughly described throughout pSHIELD study (2302).

Miscellaneous/Secure functionalities
Functionalities should be performed in a secure way: firmware upgrade, boot (01005, 01006).

5.2 System Architecture Privacy Requirements

Privacy
Data should be accessed only by authorized users (0303).

Privacy at Power Node
The FPGA engine should include a core logic that encrypts any sensitive data prior to transmitting it across the network or storing it on the embedded storage. Also, in the privacy chain BIOS password protection should be included (09009, 09010).

Asymmetric cryptography

A node should have a HW implementation of asymmetric cryptography (01033).

5.3 System Architecture Dependability Requirements

Integrity
The system should be able to prevent improper alterations (20043).

ESs integration/expansion
In case of addition of new ESs in the system, it should be possible to easily evaluate the impact of the modification on the overall system dependability (0305).

Mechanism for failure mitigation
The pSHIELD system should provide robust mechanisms to mitigate the effects on the system of the following logical threats: software failures, hardware failures, transmission failures (0307, 0308).

Trusted and dependable connectivity
The pSHIELD system shall allow trusted and dependable connectivity (20014).

Usability of ESs devices
Middleware of the pSHIELD system should enable any embedded device to be usable from a pSHIELD application (20025).

Availability
A pSHIELD node should be designed with mechanisms that improve system availability (e.g. middleware service discovery functionality should list node and network services) (01014, 0703).

Safety
Node design should comply with safety standards. Also, the node should have a safe state, in which the harmful consequences of a failure are minor (01016, 01020).

Operational states of a node
They should be ACTIVE, LOW-POWER STANDBY and TEST (01021).

Self test of nodes
Nodes of all types should be able to execute self test functionalities (01022, 09015, 06014).

Power
Nodes should present the following attributes regarding Power: uninterruptible supply, supply monitoring, supply fault tolerance, remote powering, protected supply (01023→01026, 09006).

Fault tolerance
Reaction of nodes in case of a fault should include: fault identification, substitution of defective node, self re-configuration (power node), error recovery, firmware redundancy and upgrade (09004, 09005, 09007, 01027, 09008, 01029, 20046).

Controlled failure
A power node should fail in a controlled way, meaning that node failures are halting and signalled (01028).

Miscellaneous/Reliable communications
Dependability requirements can be also reflected to a set of attributes concerning the establishment of a trustworthy network. These properties are more communication prerequisites than strict dependability requirements and cover aspects like the following: system and network homogeneity/heterogeneity, gateway node, satellite positioning antenna, heterogeneous communication support, network interoperability, robust and reliable communication links, SPD transmission, protocol conversion, proxy running on a dedicated gateway, TPM, SNMP interface (20001, 20002, 20006, 20007, 20008, 20009, 06001, 06002, 20013, 20023, 20024, 01041).

6 System Design

6.1 Node Layer

The pSHIELD Node Layer concerns the creation of an Intelligent ES HW/SW Platform encompassing intrinsic, innovative SPD functionalities, providing proper services to the pSHIELD Middleware Adapters enabling the pSHIELD Composability and consequently the desired system SPD.

Three different kinds of Intelligent ES Nodes are expected:

- nano nodes
- micro/personal nodes
- power nodes

These three node types (which can be considered three node levels of increasing complexity) represent the basic components of the lower part of the SPD Pervasive System and cover the possible requirements of several market areas.

Each of these node types provides different levels of capabilities to the remaining pSHIELD layers.

This section covers the specifications of the pSHIELD SPD Node Layer, and provides a high level description of the major components and interfaces that make up SPD Nodes. It then describes a general architecture for building-up pSHIELD SPD Nodes and some examples of specific architecture decisions for different kinds of Nodes.

6.1.1 Formal conceptual model

The figure below provides a conceptual model of a pSHIELD Node Layer. This is a generic model for all the pSHIELD Node types, which can be implemented in different architectures, providing different functionalities, different SPD compliance levels and different capabilities, depending on the type of node and application field.

SPD Node architecture is composed of different functional blocks and each one can implement several features of different complexity and performance. The choice of which features to implement depends on the application scenario and SPD Compliance Level required for the system.

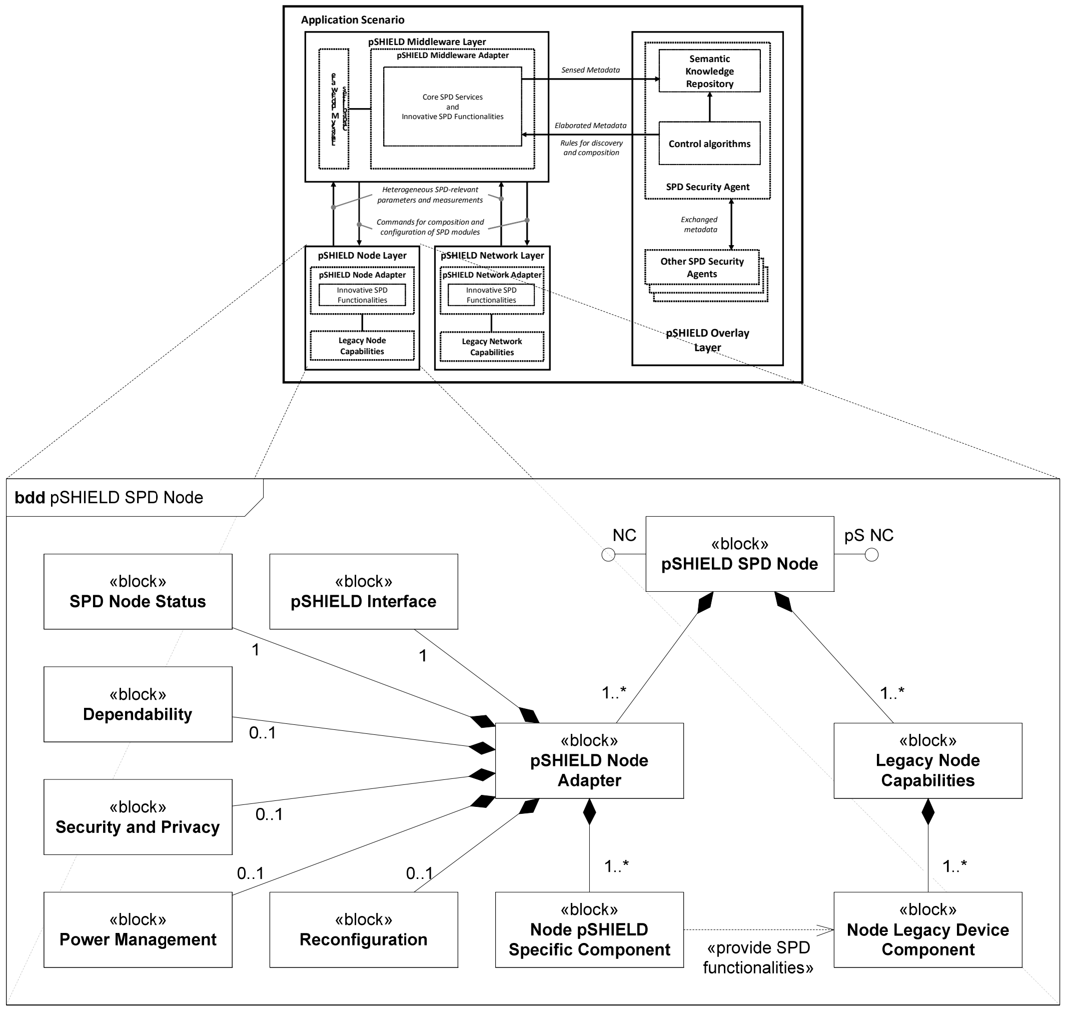

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Exchanged metadata
Other SPD Security Agents
pSHIELD Overlay Layer
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
bdd pSHIELD SPD Node
NC
«block» pSHIELD SPD Node
pS NC
«block» SPD Node Status
«block» pSHIELD Interface
«block» Dependability
«block» Security and Privacy
«block» Power Management
«block» Reconfiguration
«block» pSHIELD Node Adapter
«block» Legacy Node Capabilities
«block» Node pSHIELD Specific Component
«block» Node Legacy Device Component
«provide SPD functionalities»
1
1
0..1
0..1
0..1
0..1
1..*
1..*
1..*
1..*

Figure 16 - Formal conceptual model of pSHIELD SPD Node Layer.

The formal conceptual model of a generic pSHIELD Node Layer can be derived from the pSHIELD functional component architecture. The pSHIELD Node Layer is an Embedded System Device (ESD) equipped with several Legacy[8] Node Capabilities and a pSHIELD Node Adapter. The pSHIELD Node Layer is deployed as a hardware/software platform, encompassing intrinsic, innovative SPD functionalities, providing proper services and capabilities to the pSHIELD Middleware Adapters to enable the pSHIELD Composability and consequently the desired system SPD.

The **pSHIELD SPD Node Layer** has two interfaces, one providing pSHIELD Node Capabilities (**pS-NC**) to the pSHIELD Middleware Layer and another with legacy, technology-dependent, Node Capabilities (**NC**).

The pSHIELD SPD Node is composed of **Legacy Node Capabilities**, which consist of one or more **Legacy Device Components**, such as CPU, I/O Interfaces, Memory, Battery, etc., and a **pSHIELD Node Adapter (pSNA)**, interacting with the legacy ESDs and providing SPD functionalities.

The **pSHIELD Node Adapter** includes a set of Innovative SPD functionalities interoperating with the legacy node capabilities in order to enhance them with the pSHIELD Node Layer SPD enabling

[8] Legacy means any third-party or of-the-shelf device component

technologies. This adapter is in charge of providing (through the pS-NC interface) all the needed information to the pSHIELD Middleware adapter to enable the SPD composability of the Node layer legacy and Node pSHIELD-specific functionalities. Moreover, the pSHIELD Node Adapter translates the technology independent commands, configurations and decisions coming from the pS-NC interface into technology dependent ones and enforce them also to the legacy Node functionalities through the NC interface.

In order to show a generic model valid both for the pSHIELD Nano, Micro, Personal and Power Nodes, the different Node Layer Innovative SPD Functionalities (i.e. SPD components) are grouped into proper **modules** containing functional subsets of the Innovative SPD capabilities provided by the pSHIELD Node. In brief, the main modules of a generic **pSHIELD Node Adapter** are:

- **pSHIELD Interface**, which provides a proper interface to the pSHIELD Network
- **SPD Node Status**, responsible for collecting the status of each individual component, and providing SPD-relevant parameters and measurements to the Middleware Layer. It also checks on system health status for self-recovery, self-reconfiguration and self-adaptation
- **Reconfiguration**, which performs module or system reconfiguration by demand of the system SPD Node Status or the Middleware
- **Dependability**, responsible for applying self-dependability at node layer, by detecting problems related to system health status, and starting recovery. It is also responsible for collecting checkpoints from the remaining pSHIELD Node Adapter modules, and retrieving this information during system recovery
- **Security and Privacy**, enforcing system security and privacy at node level, by providing hardware or software encryption, decryption, key generation, firmware protection, etc.
- **Power Management**, module for managing power sources, providing protection against blackouts, etc.
- **Node pSHIELD Specific Components**, which are the innovative SPD functionalities provided to each of the Legacy Device Components, such as status and metrics, checkpoint-recovery, etc.

Depending on the type of node, application, technology, etc. each of these modules may be implemented with different pSHIELD SPD functionalities or even be not implemented. More information on each of these modules is provided in Section 6.1.1.4.2.

The pSHIELD Node may be supported by generic hard boards with CPUs or PICs and FLASH memory, special designed boards, boards with FPGA (partially dynamically reconfigurable or not), etc.

Section 6.1.1.4.1 describes some different configuration possibilities for these boards, depending on node types.

6.1.1.1 Description of pSHIELD SPD Node Layer Blocks

6.1.1.1.1 Node Legacy Device Component

This may be any legacy, third-party or of-the-shelf component. Examples are: single or multi-core CPU, network adapter, Input/Output device, sensor, actuator, memory, power device, etc. By themselves, these components are not SPD compliant, meaning that they don’t expose any capability to the Middleware, and need the corresponding Node pSHIELD Specific Components to do so.

6.1.1.1.2 Node pSHIELD Specific Component

Each Node Legacy Device Component must have a corresponding Node pSHIELD Specific Component that provides SPD capabilities. This component is mandatory for each legacy device. However, depending on node level, some of these capabilities may be optional. The list of capabilities that must or can be provided to each legacy device component is:

- Proxy – providing an interface between the device and the middleware layer, by providing the necessary elements, such as its functionality, an ID, composability information, etc.
- Status – providing the device status to the SPD Node Status through a periodic heartbeat. If an error is detected in the device, depending on its severity, this specific component may either send the error data inside the status information or stop sending the heartbeat. If possible, any actions from this specific component and Device are disabled, preventing error propagation
- Checkpoint – the internal status of both the Legacy Device and Specific components are also sent to the Dependability block for checkpointing
- Rollback-recovery – on system recovery, the Specific Component should be able to recover the component status, stored at stable storage (SS), and restart it
- Self-test – the Specific Component may also perform its own monitoring activities, such as performing a Power-On Self-Test on the Legacy Device Component

All the other modules of the pSHIELD Node Adapter expose the same capabilities.

6.1.1.1.3 SPD Node Status

This block supervises all other blocks at Node level, by collecting their periodic status information. A Heartbeat containing the global layer status is sent to the Dependability block. If one of the blocks fails the periodic heartbeat with status information or receives an error, the SPD Node Status block also stops its own heartbeat, and the Dependability Block starts recovery (e.g. by resetting the system).

This global status information may also be sent to the overlay layer.

The SPD Node Status may also send extended status information to dependability block, for a possible post-mortem analysis.

All the other blocks at Node layer must send periodic status information to this one. Invalid or inexistent status information is considered as block failure.

This block may also perform its own monitoring activities, such as performing a Power-On Self-Test.

6.1.1.1.4 Dependability

This block supervises all other blocks at Node level, by collecting their periodic status information. A Heartbeat containing the global layer status is sent to the Dependability block. If one of the blocks fails the periodic heartbeat with status information or receives an error, the SPD Node Status block also stops its own heartbeat, and the Dependability Block starts recovery (e.g. by resetting the system).

This global status information may also be sent to the overlay layer.

The SPD Node Status may also send extended status information to dependability block, for a possible post-mortem analysis.

All the other blocks at Node layer must send periodic status information to this one. Invalid or inexistent status information is considered as block failure.

This block also maintains a stable storage for storing checkpoint information from every other block, or from the middleware layer. This data is retrieved on request, for system recovery. Stable Storage[9] is a memory whose contents survive system malfunctions. It must resist to external and internal failures, and guarantee atomic reads and writes. It is usually implemented using two memory banks and when data is send to this memory, it is first written in bank1, then internally into bank2 and, only when both operations are completed, the stable write is considered done. Some error detection codes are also added, i.e. ECC, EOS, Parity algorithms. A stable read consists of checking parity bits and comparing both banks. Only when both banks are corrupted or are different from each other, but both with valid codes, the read fails. A simpler version could be simply the addition of error detection and correction codes. The control of stable writes and stable reads may be performed by hardware or software. Depending of the stable storage control, the HSM may consist of hardware, software or both. Every detected error (tolerated or not) should be stored for informing the SHSM. These features are necessary to implement the rollback recovery strategy.

6.1.1.1.5 Reconfiguration

This block is responsible for node reconfiguration. The reconfiguration of the system is made by request, usually from middleware, or may also be requested by the SPD Node Status block, as a self-reconfiguration capability (e.g., a certain threshold of a block's health has been reached).

Reconfiguration may consist on connecting/disconnecting some devices, or reprogramming the FPGA, either totally or partially.

6.1.1.1.6 Security and Privacy

This block is responsible for providing security and privacy related capabilities, such as Data Encryption, Data Decryption, Generation of Cryptographic Keys, etc.
These capabilities may be used both by other modules (such as dependability, for encrypting stable storage data) or by the middleware layer.

6.1.1.1.7 pSHIELD Interface

This block provides a proper physical interface to the pSHIELD network. It also provides the necessary drivers and capabilities to the upper levels.

6.1.1.1.8 Power Management

The Power Management (PM) module is responsible for managing power supply, such as switching to redundant power sources or checking the level of power consumption.

6.1.1.1.9 pSHIELD Node Layer Capabilities and Functionalities

A pSHIELD Node must provide to the other layers of the pSHIELD framework a set of Node Layer Innovative SPD Functionalities that comply with the pSHIELD conceptual model.

The Node Layer capabilities and functionalities are provided by the hardware, firmware or software, in the form of device drivers or other interfacing modules. The legacy node capabilities are always provided through the pSHIELD Node Adapter, in the form of pSHIELD Node Capabilities.

This section describes the pSHIELD SPD functionalities and capabilities provided by the pSHIELD Node Layer.

[9] "Software-Implemented Stable Storage in Main Memory", João Carlos Cunha, Dep. Engenharia Informática e de Sistemas, Instituto Superior de Engenharia de Coimbra; Centro de Informática e de Sistemas da Universidade de Coimbra Coimbra, Portugal; João Gabriel Silva, Dep. Engenharia Informática, Universidade de Coimbra, Centro de Informática e de Sistemas da Universidade de Coimbra Coimbra, Portugal IX Brazilian Symposium on Fault-Tolerant Computing (SCTF), Florianópolis/SC, Brazil, March 2001

6.1.1.1.10 Security and Privacy SPD functionalities

- **Encrypt/Decrypt data** – allows the encryption and decryption of data for local storage, transmission over the network or even communication with other peripherals.
- **Secure Firmware Upgrade** – allows secure firmware upgraded either locally or remotely, for system configuration
- **Cryptographic Keys generation** – allows the generation and storing of cryptographic keys for security and privacy operations.
- **Login/Logout** – allows a user to login or logout either locally or remotely

6.1.1.1.11 Dependability SPD functionalities

- **Stable read/stable write** – reads and writes data, e.g. a checkpoint, in stable storage
- **Reconfigure** – requests reconfiguration of the system. This reconfiguration may be the connection or disconnection of a device, the reconfiguration of an FPGA, etc.
- **Recover** – Requests recovery of the system from failure. This recovery may be partial (a module, a block from the FPGA, only software, etc.) or total (e.g. write full bit-stream in the FPGA and restart system)
- **Fail safe** – requests system to go to a safe state and stop
- **Self-test** – requests for a partial or full self-test of the system
- **Degrade functionality** – requests a system reconfiguration to function in a degraded mode, e.g. for power saving
- **Degrade dependability** – requests a system reconfiguration to decrease dependability, e.g. after failure of a redundant module

6.1.1.1.12 Performance/Metrics SPD functionalities

- **Get performance/metrics** – gets performance and metrics information from the whole system

Following, the list of the metrics provided by the node:

1. System and components health status
2. System and components configuration
3. Power consumption
4. Power supply status
5. Number of detected errors per type and component
6. Number of recoveries per types and component
7. Failed components
8. Number of intrusion attacks

6.1.1.1.13 Discovery/Composability SPD functionalities

- **Discovery** – provide to the pSHIELD Middleware Adapter the information, raw data, description of available hardware resources and services in order to allow the system composability
- **Connect/Disconnect** – connects or disconnects specific SPD functionalities for system composability

6.1.1.1.14 Miscellaneous SPD functionalities

Depending on the application field, other services are provided, mainly related to the Special Purpose Processor modules:

- **Compress/decompress** – requests data compression or decompression for local storage or exchange over the network or with peripherals
- **Configure/calibrate** – requests the configuration or calibration of a device attached to the node
- **Digital Signal Processing** – digital signal acquisition and conversion (ADC/DAC)

6.1.1.1.15 Node Status

- **Node Layer Status** – provide to the pSHIELD Middleware Adapter the status information from each of the modules in Node Layer

6.1.1.1.16 Legacy SPD functionalities

Legacy devices provide SPD functionalities or capabilities through the pSHIELD specific components. Some examples are:

- **Compress/decompress** – requests data compression or decompression for local storage or exchange over the network or with peripherals
- **Configure/calibrate** – requests the configuration or calibration of a device attached to the node

  Digital Signal Processing – digital signal acquisition and conversion (ADC/DAC)

6.1.1.2 Intrinsic SPD capabilities

To comply with SPD requirements, the pSHIELD Node has some Node Layer intrinsic, architectural characteristics specially designed to provide dependability, security and privacy exclusively at Node layer.

This intrinsic SPD capabilities can be configured by the pSHIELD Overlay and composed by the pSHIELD Middleware Core SPD Services, however they apply autonomously and continuously in the pSHIELD Node.

6.1.1.2.1 Dependability

Dependability is mainly assured by the Dependability block, the SPD Node Status block and by all the functionalities embedded in the other blocks, such as those described in the Node pSHIELD Specific Component, providing the status of the module, checkpointing information, self-test and rollback-recovery.

If any error is detected by any of the modules, the Dependability block triggers system recovery. The Dependability block itself must be able to detect a self-failure, by having a redundant component, such as a watchdog timer that starts system recovery.

Other modules also provide other aspects of dependability, such as the Power Management (power failures).

There are thus several levels of dependability:

- Each module of the pSHIELD Node Adapter has an internal Health Status Module (HSM) that monitors its health and periodically sends health status information to the SPD Node Status. The SPD Node Status sends a periodic Heartbeat containing the global layer status to the Dependability module
- On error, the HSM may inhibit the monitored module, performing a fail-fast operation. If the SPD Node Status stops receiving status information from one of the HSM or receives error information or even the information itself is erroneous, it stops the Heartbit. On timeout, the Dependability module starts a recovery procedure
- The SPD Node Status may also perform other health status monitoring operations, such as performing a Power-On Self-Test
- Dependability module itself sends a health status information to the SPD Node Status
- If the Dependability module fails, the SPD Node Status halts the system
- On permanent failure of one of the modules, the Dependability module may halt the system
- The Power Management assures system availability by managing redundant power sources or triggering a low-power mode if power level is low
- The Dependability module contains a Stable Storage, assuring data survivability for rollback-recovery

The pSHIELD power node may exhibit advanced recovery and reconfigurability capabilities through partial FPGA reconfiguration[10]. Recent advances in FPGA technology offer the possibility of repairing a failed module by reloading the bit stream in the FPGA frames that contained this module[11]. Furthermore, this FPGA reconfiguration may be used for changing the device functionality during runtime.

[10] "In-Circuit Partial Reconfiguration of RocketIO™ Attributes", http://www.xilinx.com/support/documentation/application_notes/xapp662.pdf

"Two flows for Partial Reconfiguration: Module Based or Difference Based", http://www.xilinx.com/support/documentation/application_notes/xapp290.pdf

"Dynamic Reconfiguration of RocketIO MGT Attributes",

http://www.xilinx.com/support/documentation/application_notes/xapp660.pdf

[11] Cheatham (portal.acm.org/citation.cfm?id=1142167)

Also depending on application criticality, other forms of fault-tolerance may be used, such as static redundancy (e.g. Triple Modular Redundancy - TMR)[12] or dynamic redundancy, such as stand-by spare. This redundancy may be applied for each one of the modules that constitute the pSHIELD Node, even Nano Node.

6.1.1.2.2 Security and Privacy

Security and privacy are assured by the Security/Privacy module. The level of security and privacy depends on the modules that are implemented, which may assure, for example, Data Encryption, Data Decryption, Generation of Cryptographic Keys, etc.

6.1.1.3 Requirements for pSHIELD Power Node, Micro/Personal Node and Nano node

6.1.1.3.1 HW/SW Implementation

A pSHIELD Node is deployed as a hardware/software platform, encompassing intrinsic, innovative SPD functionalities, providing proper services to the other pSHIELD Network and Middleware Adapters to enable the pSHIELD Composability and consequently the desired system SPD[13].

The three kinds of **pSHIELD SPD Node** each deploy a different configuration of Node Layer SPD functionalities of the pSHIELD framework and comprise a different type of complexity: **Nano nodes**, **Micro/Personal 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.

The table below presents typical hardware deployed in each node type for every module of the pSHIELD SPD Node conceptual model.

[12] "On the Reliability of Cascaded TMR Systems", Masashi Hamamatsu, Nomura Research Institute, Ltd., Yokohama-City, Japan, Tatsuhiro Tsuchiya Tohru Kikuno, Osaka University, Suita-City, Japan, 2010 Pacific Rim International Symposium on Dependable Computing

[13] "Security and Dependability of Embedded Systems: A Computer Architects' Perspective" Jörg Henkel, University of Karlsruhe, Karlsruhe, Germany; Vijaykrishnan Narayanan, Pennsylvania State University, USA; Sri Parameswaran, University of New South Wales, Australia; Roshan Ragel, University of Peradeniya, Sri Lanka VLSID '09 Proceedings of the 2009 22nd International Conference on VLSI Design EEE Computer Society Washington, DC, USA ©2009

pSHIELD SPD node block	Power node	Micro/Personal node	Nano node
Application Processor (Legacy Device Component)	Multi-core processor (HW and/or SW core)	Microcontroller (HW and/or SW core)	Microcontroller
Non-volatile memory (Legacy Device Component)	ROM, EEPROM, FLASH, Hard Disk or other forms of non-volatile memory	ROM, EEPROM, FLASH	ROM, EEPROM, FLASH
Volatile memory (Legacy Device Component)	RAM, SRAM, DRAM	RAM, SRAM, DRAM	RAM
Special-Purpose Processor (Legacy Device Component)	Hardware digital signal processing (DSP) and/or glue logic blocks	ADC	ADC
Node pSHIELD Specific Component	Hardware or Software	Hardware or Software	Hardware or Software
Stable Storage (Dependability Block)	Flash-based, 2 memory banks, HW or SW control	Flash-based, single memory bank	N.A.
Reconfiguration and Recovery Controller (Dependability Block)	IP Core	IP Core, ASIC or VLSI	N.A.
I/O Interface	USB, ETHERNET, UART, CAN, RS232, RS485, GPIO	GPIO,ETHERNET, RS232, CAN	SERIAL; Wi-fi; RF-ID;BT; Zigbee;
Power Management	UPS, Power Monitoring Device	Power Monitoring Device	N.A.
Security and Privacy	AES Encryption TPM Module	TPM module OTP (one time) Password	N.A.

Table 1 - pSHIELD enabling technologies by node types.

6.1.1.3.2 Capabilities and Functionalities

As previously stated, different pSHIELD node types are enabled by different technologies and provide different functionalities.

Depending on the Node Type, different capabilities and functionalities, such as those described in Section 6.1.2, may be Available [A], Not Available [N.A.] or Optional [O.], such as presented, as example, in the following table:

	pSHIELD Node Type		
pSHIELD Node SPD functionalities	**Power node**	**Micro/Personal node**	**Nano node**
Security/Privacy functionalities			
Encrypt/Decrypt data	[A]	[A]	[N.A.]
Secure Firmware Upgrade	[A]	[A]	[O]
Cryptographic Keys generation	[A]	[A]	[N.A.]
Login/Logout	[A]	[A]	[A]
Dependability functionalities			
Stable read/stable write	[A]	[O]	[N.A.]
Reconfigure	[A]	[N.A.]	[N.A.]
Recover	[A]	[A]	[N.A.]
Fail safe	[A]	[O]	[N.A.]
Self-test	[A]	[O]	[O]
Degrade functionality	[A]	[O]	[N.A.]
Degrade dependability	[A]	[O]	[N.A.]
Power Management			
Change power	[A]	[O]	[N.A.]
Performance/Metrics SPD functionalities			
Get Performance/Metrics services	[A]	[A]	[N.A.]
Discovery & Composability functionalities			
Discovery	[A]	[A]	[A]
Connect/Disconnect	[A]	[A]	[A]
Node Status			
Node Layer status	[A]	[O]	[N.A.]
Legacy SPD functionalities			
Compress/decompress	[A]	[O]	[N.A.]
Configure/calibrate	[A]	[O]	[A]
Digital Signal Processing	[A]	[N.A.]	[N.A.]

Table 2 - pSHIELD services by node types.

6.1.2 Nano and Micro/Personal node HW/SW

The technology advancements in computing hardware and software enables a new generation of small Embedded System Devices (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 Nano, Micro/Personal Nodes is two fold:

1. To design completely new nano, micro/personal nodes that are complaint with the pSHIELD system design as described in Section 6.1.1
2. To keep legacy technologies as they are, developed for many applications including those that are targeted in pSHIELD, which means to assume 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 nano, micro/personal node equipped with some Legacy functionalities and with the pSHIELD Node Adapter, 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 pSHIELD Node keeps almost all desired functionalities of a standardised sensor technology with additional SPD features that make it composable into the pSHIELD framework. The architectural design of the pSHIELD Nodes will relay on the ISO/IEC 9126 standard that has 6 top level characteristics: functionality, reliability, usability, efficiency, maintainability and portability.

The architectural design of the pSHIELD Nodes is not an easy architectural task since it requires facing many different constrains in the same time. Some of these constraints can converge in the same direction but some of them will be divergent and in the opposite directions. To cope with this challenge architectural design, as shown in section 5, the pSHIELD ESD use two approaches: (i) a Network approach and a (ii) Functional approach. The network approach constrains the architectural system design from network point of view. This approach should guarantee that all pSHIELD Nodes are part of a SPD Network that can be easily integrated with standard IP-based network like GSM, UMTS, etc. In other words it means that an SPD network is implementable and interoperable with standard networks to comply the main business cases of the application scenarios. The Functional approach constrains architectural design from the SPD requirements related to the node, network and middleware layers. The real innovation of pSHIELD is the introduction of the Overlay that makes the double approach to converge.

6.1.2.1 Nano, Micro and Personal node description

The following figure provides a general view of the pSHIELD Nano, Micro/personal Node architecture. This is a generic model, which can be implemented in different architectures, providing different functionalities and different services, depending on the tasks to be accomplished by the node and the application field.

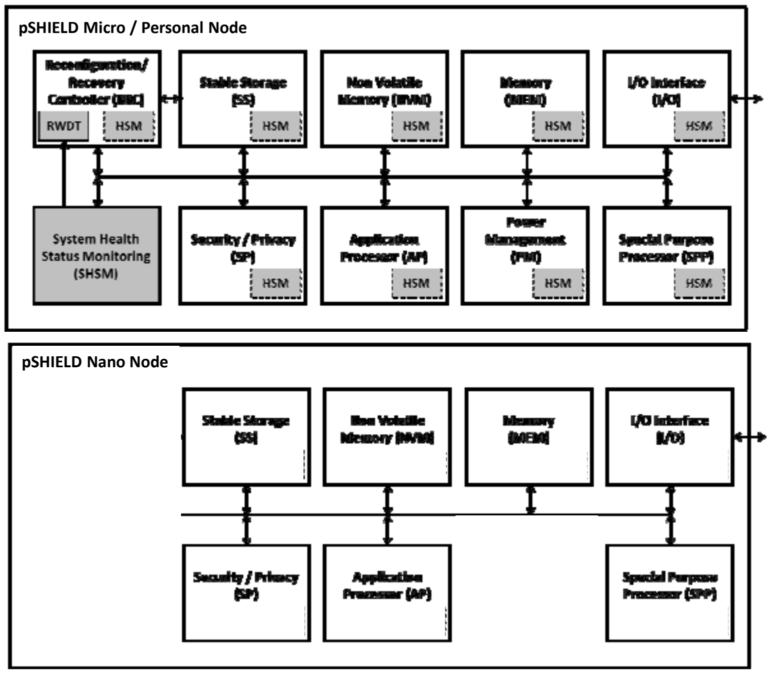

pSHIELD Micro / Personal Node
Reconfiguration/ Recovery Controller (RRC)
RWDT
HSM
Stable Storage (SS)
Non Volatile Memory (NVM)
Memory (MEM)
I/O Interface (I/O)
System Health Status Monitoring (SHSM)
Security / Privacy (SP)
Application Processor (AP)
Power Management (PM)
Special Purpose Processor (SPP)
pSHIELD Nano Node

Figure 17 - Schematic view of SPD modules of a generic pSHIELD Nano, Micro and Personal Node.

Health Status Monitoring and Controllers take care for different control functionalities. The hardware interface will cover the specification of the cryptographic hardware security blocks, high demanding level security performance, for example secure boot, secure time-stamping and all necessary security management functionality such us device administration, key creation and key import-export. Additionally, it defines the hardware interpreted data structures and direct interdependencies.

	High Security Level	Medium Security Level	Low Security Level
MEM	64 kByte	64 kByte	Optional
NVM	512 kByte	512 kByte	Optional
SPP Cryptography	AES-128 CCM, GCM, ECC	AES-128 CCM, GCM , ECC	AES-128 CCM, GCM, ECC
SP	AES-PRNG with TRNG seed	AES-PRNG with TRNG seed	Optional
AP	ARM Cortex-M3 32 bit, 50– 250 MHz	ARM Cortex-M3 32 bit, 50– 250 MHz	No
I/O Interface	Yes	Yes	Yes

Table 3 - An example of the personal node SPD components.

From the above figure we can see that sensor, memory, radio and interface units are more or less standard modules for any standardized sensor technology. The multi-core processor will play a key role of the ES design with SPD features. For example, the memory can be realized by ROM, RAM, and FLASH, the radio can be 802.15.4, 802.11, RFID, UWB, etc. and the interfaces can be ADC and DAC, Timer, UART, I2C, etc. Finally, the multi-core processor can be realized as single microcontroller for nano and micro nodes or more than one core microcontroller for personal nodes and multi-core processor for power node (SPD & HSM, Application and Specific processors).

Today 3D integration nano-technology offers a new perspective for extremely complex heterogeneous System On Chip (SoC) design. The following figure shows hardware architecture of SoC design.

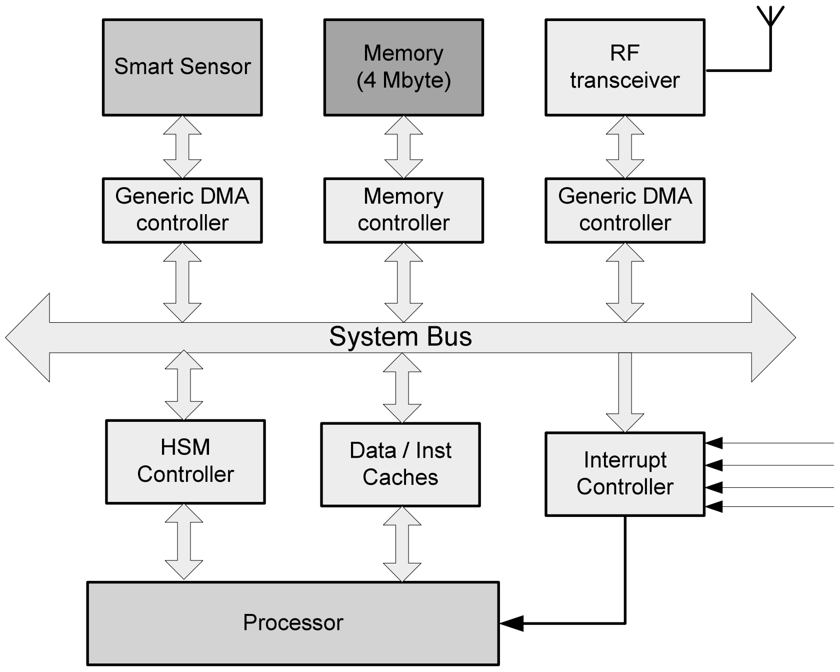


Figure 18 - Hardware architecture and nano node chip partitioning.

To address the innovative issues and challenges in pSHIELD, solutions and **long-term objectives** are proposed for the development of pSHIELD Nano Nodes:

- A new hardware architecture described in above figure is proposed based on two Innovative SPD components:
  - an intelligent low power smart sensor with on chip detection capability
  - a digital image processing and communication chip based on optimized signal processor
- Low power nano node with a target of less than 1mW by defining and implementing a multi-level power management strategy
- Miniaturization through 3D Integration with special care for thermal study and electrical interactions between analogue, digital and RF
- Autonomous system working on a battery and communicating an optimized dataflow through wireless RF link

For a typical multi-task software application running on a mono-processor architecture (MIPS32 processor core with separated data and instruction caches), the ultra-low power processor chip can be designed by using VHDL (Very high speed integrated circuits Hardware Description Language). It contains also many digital peripherals like timers, watchdogs, interrupt controllers, HSM controller, UART (Universal Asynchronous Receiver/Transmitter), SPI (Serial Peripheral Interface), DMA (Direct Memory Access) controllers and interfaces/controllers to memories and cache memories. The design will also be performed taking into account the constraints of 3D integration.

6.1.2.2 Nano, Micro and Personal Node operating systems

Selection of the operating system for the demonstrator is an important design constrain, since we need to decide in which sensor platform SPD functionalities will be realized. The only requirement that we posed for this operating system is related to the possibility to be designed for embedded devices. There are two candidates for that: TinyOS and Contiki. Additionally, Hydra platform is a new concept that is realized in such a way that between physical and application layer is only a middleware.

6.1.2.2.1 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 optimized for the memory limitation of sensor networks. TinyOS features in summary:

- No Kernel: Direct hardware manipulation
- No Process Management: only one process on the fly
- no Virtual Memory: Single linear physical address space
- No S/w Signal or Exception: Function call instead
- No User Interface, power constrained
- Unusually application specific HW and SW
- Multiple flows, concurrency intensive bursts
- Extremely passive vigilance (power saving)
- Tightly coupled with the application
- Simulator: TOSSIM, PowerTOSSIM
- Written in “nesC” Language, a dialect of the ‘C’ language

6.1.2.2.2 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. Contiki features in summary:

- Event-driven Kernel: reduce the size of the system
- Preemptive multi-threading support: an application library that runs on top of the event-driven kernel is optionally linked with applications that explicitly require a multithreaded model of computation
- Simulator: COOJA
- Written in ‘C’ Language

6.1.2.2.3 HYDRA middleware

The European Hydra project developed a "Middleware for Heterogeneous Physical Devices" with the aim to help manufacturers and systems integrators to build devices that can be networked easily and flexibly to create cost-effective high performance solutions. For the heterogeneous devices, sensors and actuators envisioned in the pSHIELD project, the large number of manufacturers and Universities are involved and the differences in their speed of innovation become an obstacle for the overall system design. Therefore, there is an urgent need for technologies and tools that make it easier to reap the benefits of networked systems. The complexity to build new technologies and tools grows exponentially with the number of devices, manufacturers and protocols involved. The Hydra middleware is a core technology that has a transparent communication layer, equally supporting centralized and distributed architectures. The Hydra middleware takes security and trust into account and allows building model-guided web services. It runs on wired or wireless networks of distributed devices with limited resources. The embedded and mobile service-oriented architecture will provide fully compatible data access across heterogeneous platforms, allowing to create true ambient intelligence for networked ESDs. Adding extended security, privacy, trust and new dependability modules may satisfy requirements for having a middleware that will be SPD composable with the rest of the pSHIELD system architecture and network. The Hydra middleware consists of large number of software components – or managers – that handle various tasks needed to support cost-effective development of intelligent applications for networked embedded devices.

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 incorporates service discovery, P2P communications and diagnostic. In Hydra framework any physical devices, sensors, actuators or subsystems can be considered as a unique web service.

What we will need from HYDRA middleware for the pSHIELD SPD nodes? A lightweight version of this middleware, to be the Legacy Middleware Layer on top of which the pSHIELD middleware Adapter can host a set of Innovative SPD Functionalities: proper software modules must be added. This solution is in line with the recent IP stacks that are lightweight enough to run on tiny, battery operated ESDs. This is also in line with emerging application space of smart objects that require scalable and interoperable communication mechanisms that support future innovations as the application space grows. This strategy is also aligned with the future application scenarios “the Internet of Human and Things” (ITH). Smart objects are small computers with a sensor and actuator and a communication device, embedded in objects. To support the large number of emerging applications for smart objects, the underlying networking technology must be inherently scalable, interoperable, and have solid standardization base to support future innovation.

6.1.2.3 Specific SPD Considerations for Wireless Sensor Networks

The Wireless Sensor Networks (WSN) applications are used in many critical tasks, like aerospace, automation, monitoring environment, etc. Nowadays, these applications include new properties, such as security, dependability, privacy and trust. For WSNs applications to make security and dependability satisfaction is more and more important. In general WSNs are layered in 5 layers, but there are also other cases like Hydra network where we have three layers. In the pSHIELD project we follow the concept of four functional layers and based on that we are constructing a new type of network that we simple call SPD network. Heterogeneity of this SPD network is an extremely important feature of the pSHIELD network, since it allows existence of different type of Embedded System Devices on the Node Layer.

In general, the software part of WSNs can be layered into three levels: sensor software, node software and sensor network software. Sensor software has full access to sensor hardware. The output of a function of sensor software is used by sensor node software. This level includes system software for network maintenance and for some specific applications. For example, middleware resides over the operating system. Application programs use this middleware according to their own specific requirements. So, bottom layer consists of sensor, CPU and radio, on top we have operating system and on top of them services and applications.

There are two approaches for sensor applications: (i) Service-oriented architecture (SOA) and (ii) Agent-oriented architecture (AOA). SOA is a design approach that defines the interaction among architectural elements in terms of services that can be accessed without knowledge of the underlying platform implementation. AOA proposes an infrastructure that applies active agent technology to WSNs, because the network must be dynamically configurable and adaptive in order to respond actively to events where security and dependability must be built into WSNs at the early design stage. The pSHIELD solution leverages this two approaches investigating a hybrid solution where the SOA is applied by the pSHIELD Middleware Adapter and AOA is applied by the Security Agents operating in the pSHIELD Overlay.

6.1.2.4 SPD models for Nano, Micro and Personal nodes

Triverdi et al. presents a classification of dependability and security model types: combinatorial models, state-space models, hierarchical models, fixed point iterative model, simulation, analytic and simulation, and hybrid model, that can be applied for the presentation of dependability and security models (below figure). For extremely difficult models analytic and simulation can be used in combination with hybrid models. Development of new SPD Embedded System Devices requires careful approach and consideration of a variety of aspects that are influencing our design methodology. Dependability and security models are developed almost independently in the area of small networked sensors. Based on the recent paper publish by Triverdi et all, called Dependability and Security Models[14] we have a solid background for modelling security and dependability for SPD ESDs.

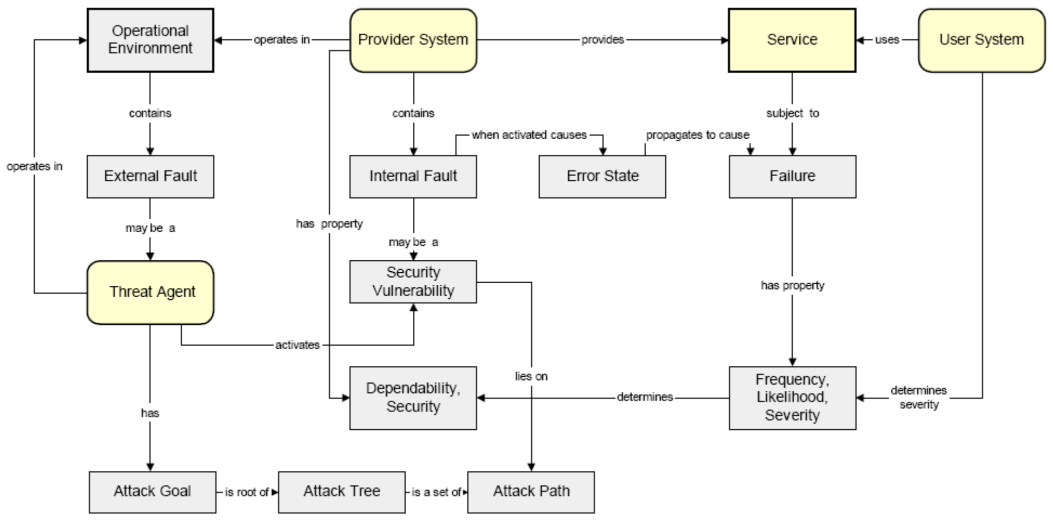


Figure 19 - Concept model for security and dependability[15].

[14] Kishor S. Triverdi et all, “Dependability and Security Models,” 7th International Workshop on the design of Reliable Communication Networks, DRCN 2009, Washington Dc, October 2009.

[15] John Murdoch, “Security measurements,” White paper, 2006.

Security is a property of a system or service*.* Software-intensive systems are complex, meaning that they are composed of many components of different types which interact with each other to create properties not exhibited by the individual components. The purpose of the system is implemented as the *service*; the system, acting as a provider, delivers to another system, the user system. A particular service can fail in a variety of ways, resulting in dependability being a composite property, covering the following more specific properties (the majority of the properties is indicative of fewer or absence of the corresponding failures). Dependability and security overlap in the sense that some types of failure fall under both properties.

The definition of dependability and security as the ability to avoid failures raises the question of how a system or service can be measured with regard to such ability. Before addressing this question, we need to define a model of how a service failure is caused.

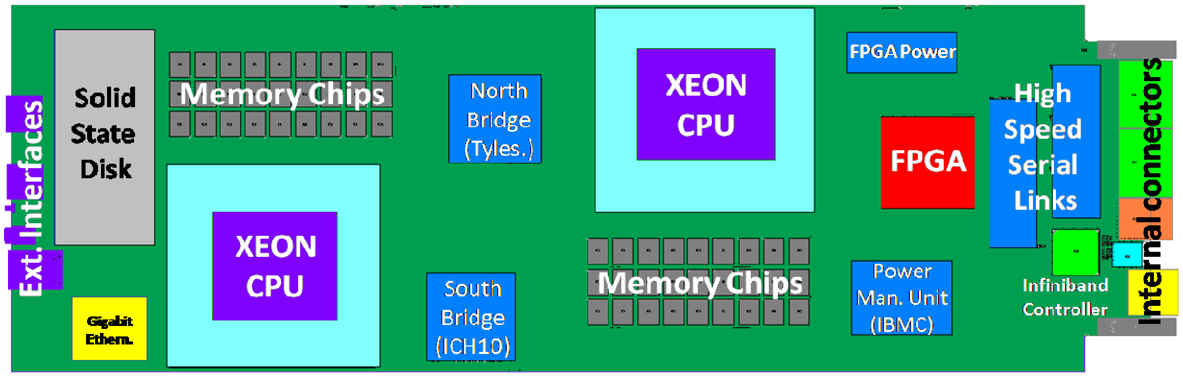
Many types of fault of concern to security system solution are similar to safety faults. For example, events in the natural environment, accidental, non-malicious actions during development etc. However, security has an additional type of fault arising from the presence of malicious threat agents in the operational and development environment. Such agents can learn and adapt, resulting in evolving external faults. Attack trees can be used to map the objectives of a threat agent onto vulnerabilities of the system. Alternative attack sequences represent the possible ways the agent might achieve his/her goal. Development and operational policies can be adjusted to prioritize defensive actions. Measurement can support the decision making involved, for example in the estimation of the cost to a threat agent of different attack sequences. Under certain assumptions, an increase in attack cost would imply a lowering of the likelihood of the attack sequence occurring and an increase in security with regard to the associated service failure.

6.1.3 Power node HW/SW

Power Node will be a rugged embedded system, optimally designed in terms of dimensions, weight, power consumption and capable to work in harsh environmental conditions. The reference application context is defence/aerospace, ground mobile and airborne environments, addressing manned and unmanned applications where reliable high performance computing is required.

The Power Node will be based on a powerful computing architecture: a dual Intel Xeon 5500/5680 series (Quad core CPU) motherboard, with at least 6GB of on-board soldered DDR3 memory and a high data retention 80GB SSD drive. A high speed, high density FPGA device will also be present, providing easy adaptability and implementation of dedicated functions and special algorithms. It will offer a maximum processing power of 80GFlops.

In the following images the concept of the Power Node is described. The first image illustrates the form factor of the Power Node board and the positioning of the components on the board itself. The second image represents the board covered by a cold-plate that can be air or liquid cooled. The shape of this cold-plate is intended, at this stage of the project, only for descriptive purposes. The final version could be different.



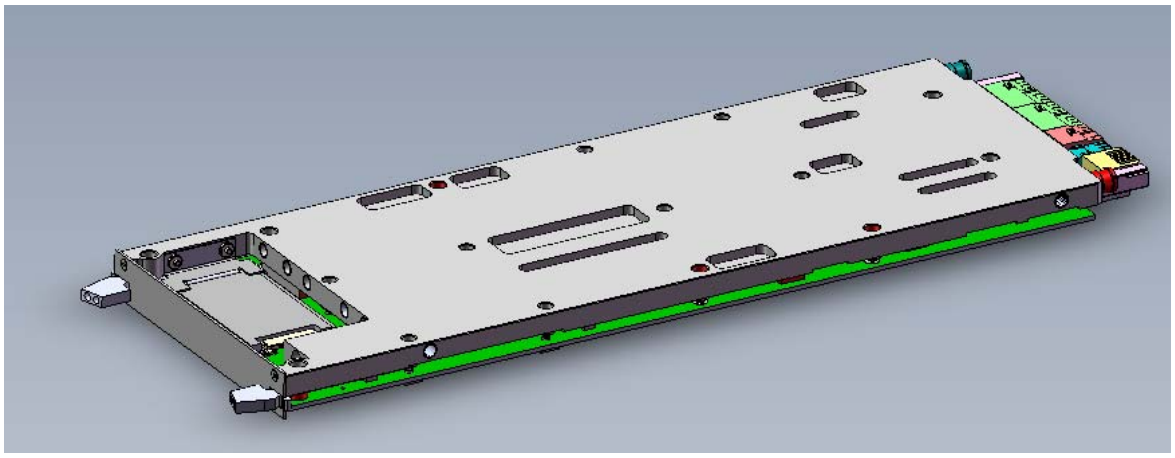


Figure 20 - Power Node board concept, with and without cooling heat sinks.

6.1.3.1 Power Node software (OS, Protocol stack, Interfaces)

The software development for the Power Node will be mainly devoted to the adaptation of a commonly available Linux Distribution, in order to benefit from the richness of the features of a widely adopted operating system.

Regarding the OS the first choice will be “RedHat Enterprise Linux OS Verison 5.5 x86_64” which needs a license but is very well supported. Alternatively, if an open-source Linux distribution is required, the Power Node can support Linux distribution derived from RedHat, which are available for free but don’t have usually an excellent support. In this case, the operating system could be one of the following:

- CentOS
- Scientific Linux

In addition to the OS the porting of device drivers for the Infiniband networking interface and for the IBMC Board Management Controller will be provided.

The design of the FPGA firmware and software is intended to be implemented by the user of Power Node using ALTERA development tools:

- QUARTUS II (http://www.altera.com/products/software/quartus-ii/subscription-edition/qts-se-index.html)
- USB-JTAG programming/debugging tool (http://www.buyaltera.com/scripts/partsearch.dll?Detail&name=544-1775-ND)

As a starting point, many reference designs, optimized for the same FPGA used in the Power Node, can be downloaded from Altera website. They reduce time to implement complex interface such as PCIe by means of pre-compiled building block.

To develop end-user applications, the final software development kit will contain the following additional tools:

- Infiniband OFED driver Stack supplied by Mellanox (basically standard OFED stack 1.5.1 pre-compiled). The package contains drivers and libraries for the InfiniBand interface and for the 10Gb Ethernet interface (http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=26&menu_section=34#tab-three)
- IPMI tools
- Scientific Computation Libraries from EPEL Repository (they need separate free licensing)
- Intel C/C++ and Fortran Compilers
- Intel Math Kernel Libraries (All the Mathematic primitives: FFT, Matrix calculations etc)
- Intel Integrated Performance Primitives (these are basically computational accelerators)
- Other Intel Libraries (these are proprietary libraries for example: treading building block)

6.1.3.2 Power Node hardware (Radio, Power, CPU, Interfaces, Sensing, extras (FPGA etc.))

The Power Node is a High Performance Platform based on Nehalem/Wesmare Xeon Intel dual-processor board with Tylesburg chipset; it is equipped with a high density FPGA and a high speed Infiniband controller, moreover there is an Ethernet Gigabit interface. Every component is supervised by a Power Management Controller Unit (IBMC).

The Power Node core architecture will consist of two Intel Xeon X5680 or X5570 CPUs, connected via Quick Path Interconnect (QPI), a dedicated low latency and high bandwidth bus capable of up to 6.4GT/s. Three channels of DDR3 memory are connected to each CPU, which integrates a high performance memory controller. The system hub (I/O Bridge) will be an Intel 5520 (Tylersburg) chipset and provides connectivity between the CPUs and the rest of the system; each CPU is connected to its Tylersburg with a QPI link. A Mellanox QDR ConnectX2 adapter is connected to the Tylersburg via one x8 PCIe 2.0 link: it provides a high Infiniband compliant connection. The hardware programmable part of the Power Node is represented by an Altera Stratix IV FPGA, which is connected to the Tylersburg with 2 x8 PCIe 2.0 links.
Finally, the peripheral hub (Intel ICH10) is connected to the Tylersburg and provides the following additional peripherals:

- one optional SATA SSD, used to provide local fast and permanent storage
- one Zoar Gigabit Ethernet adapter
- 2x external accessible USB ports
- one Output Video Port

- one UART for low level debug

The independent, embedded controller for the Power Management (IBMC) allows the monitoring of each performance parameters, such as temperatures, voltages, etc. Access to these parameters can be done by the Power Node applications, locally and remotely over the network. The IBMC provides an SNMP interface to the Power Node and allows setting traps for specific events. It can also trigger and monitor the Power-On-Self-Test. In terms of remote control, the embedded IBMC permits the remote configuration of the Power Node through the network and additional remote configurability can be done through the FPGA.

The overall architecture of the Power Node is represented in the next figure.

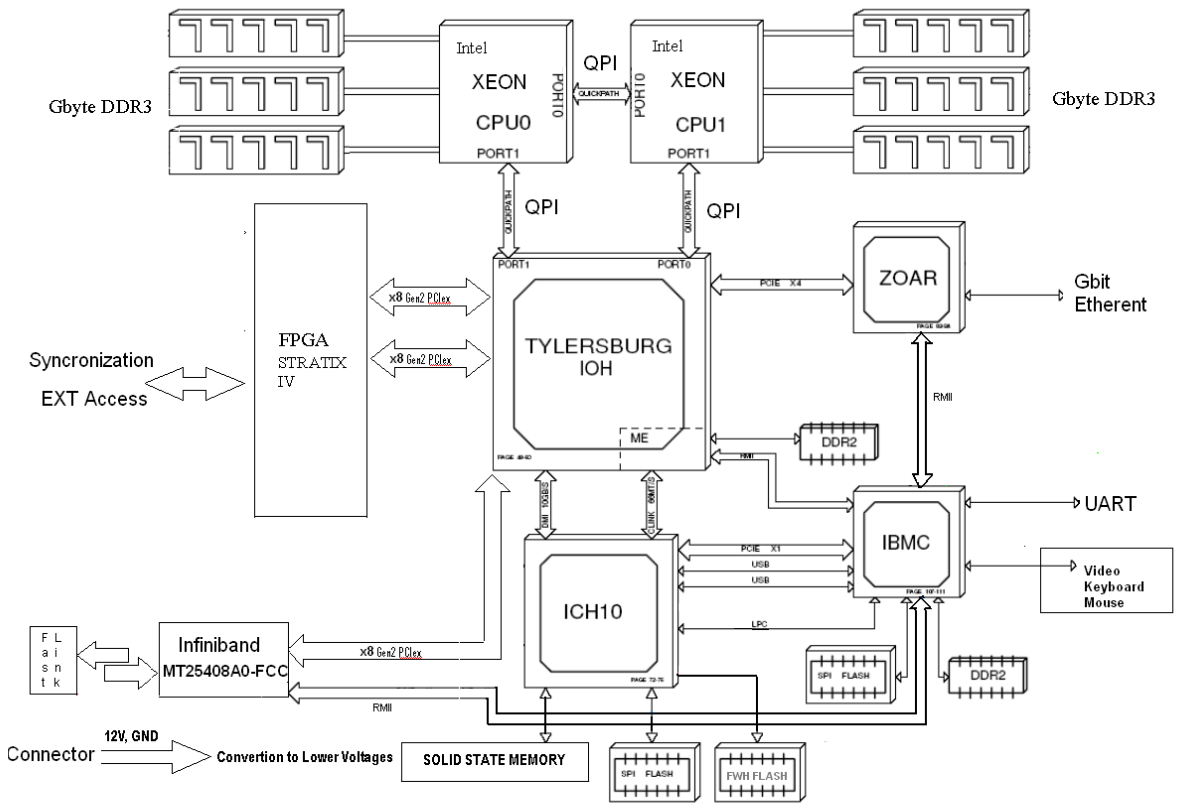


Figure 21 - Power Node architecture: high level description.

The FPGA Processor is responsible for some security aspects. It includes a core logic that monitors the security of the Power Node. Tampering with the node triggers a protection mechanism in the security node that:

- physically disconnects any I/O and network
- deletes any data resident on the node
- initiates the physical destruction of the device itself by driving the power supply

- provides security features such as cryptographic capabilities through a dedicated core embedded in the FPGA
- more in general, the hardware supports the Intel AES-NI technology

The Power Node architecture has been conceived thinking also “composability”, in order to provide the possibility to build network of Power Nodes depending on the specific requirements of the specific application context. The Infiniband interface allows creating virtual 3D torus networks of Power Nodes, which are very efficient in terms of bandwidth and latency, and are capable of scaling up with no performance penalty. The torus network is managed by a network processor implemented in the FPGA of each Power Node, which interfaces to the system hub through two x8 PCI Express Gen 2 connections, for a total internal bandwidth of 80Gbs. Thanks to the FPGA implementation, the torus network processor permits standard, ad-hoc and application-dependent collective communications. Finally, the I-O and network interfaces are programmable, in order to permit interfacing the system to multiple network and bus technologies and protocols, increasing in this way the potential scalability of the network.

The possibility to aggregate multiple identical units has an impact also on dependability, providing redundancy. The execution segregation through hardware virtualization allows for protection, monitoring, disabling and replacement of malfunctioning or compromised nodes. Moreover, in case of a fault, redundant hardware provides dependable operations. This is accomplished at the hardware level through duplication of the resource and at a functional level through aggregation of resources (spare Power Nodes).

6.1.3.3 Power Node Reconfigurability

The capability of the Power Node to reconfigure itself, at runtime, is offered by the use of “in-system programmable” devices such as an FPGA. This means that according to an environmental request, not only the software libraries can be dynamically loaded, but also the hardware accelerator configuration can be modified at any time. With configuration we intend the hardware logic previously programmed in the FPGA.

As shown in the following image, hardware silicon internal part of the FPGA are based on SRAM Logic Elements which consist of combinational logic attached to memory elements and they can be combined to implement any type of hardware function.

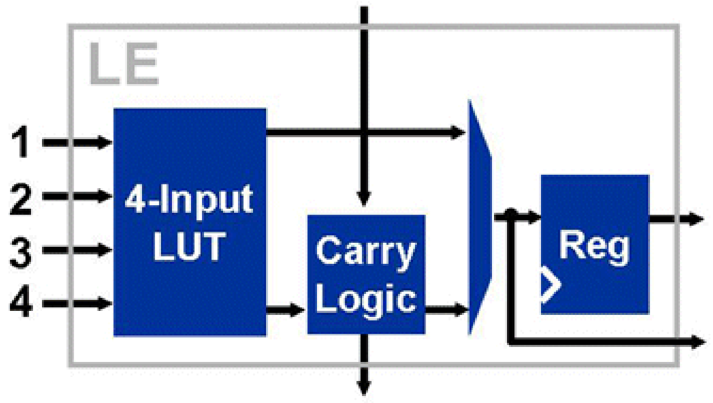


Figure 22 – Internal structure of an FPGA logic element.

Complex hardware functionalities can be designed with high level hardware description languages such as VHDL or Verilog or through schematic entry tools provided by development IDE.

Once the design has been completed and synthesized, the development tools provide a binary file which can be written to the target device (FPGA) to update the configuration to the newer one.

The standard interface to access configuration registers of the FPGA is the JTAG port and it is used to write on it the binary file produced by the compiler.

The Power Node uses a USB-JTAG converter to grant OS the access to HW reconfiguration. The converter is integrated on the Power Node board. This solution has been adopted on both release of the prototype to simplify and improve the development and debug process. A second solution, that doesn't require the USB-JTAG converter, could be adopted in future versions of the prototype that will be closer to a final product. The current hardware already allows the implementation of this solution that, in terms of functionalities, is perfectly equivalent to the one adopted. This second solution is based on the direct reconfiguration of the FPGA through the PCi Express bus. A software application is capable to store the FPGA binary images into the Flash memory connected to the FPGA, and chooses the most suitable image depending on the threat identified. In this case, a specific operating system driver must be implemented to control the PCi Express bus and an engine that acts as a bridge between the bus itself and the flash memory, must be implemented into the FPGA and added to the FPGA application specific logic.

The reconfigurability features offered by the Power Node can be used in a real application scenario as follows:

1. A threat is identified by proper application logic
2. The application, depending on the threat, decides if a reconfiguration of the FPGA is required
3. The operating system stops processes that use the current hardware configuration
4. The application chooses the new configuration capable to face the threat
5. The selected configuration is written via JTAG to the FPGA

The operating system starts new processes associated with the new HW configuration.

6.1.3.4 Technical Specifications

In terms of technical specifications, the Power Node will feature:

- 2 Intel Xeon 5570/5680 CPUs at 2.93/3.33GHz
- At least 6GB RAM 1333MHz DDR3
- Optional FPGA device, which allows implementation of:
    - ✓ Hardware accelerator features (on board co-processing)
    - ✓ Synchronization network for multi node mode
- Custom processing units
- Optional 80GB 1.8" SATA SSD
- Independent sensor network and monitoring system
- Connectivity via two additional debugging board that will bring the signal on standard connectors for an easier access by the user
- QDR Infiniband port
- LAN 10/100/1000 Interface
- VGA Analog Video output
- 2x USB 2.0 host interface

Physical Specifications:

- Physical dimensions: 166mm h x 25,4mm w x 500mm d
- Weight: 2.2 Kg (with cooling system)

Power Specifications:

- Power consumption: 350W typical (420W Max)
- Power Supply Voltage 12V

6.2 Network Layer HW/SW

6.2.1 pSHIELD Network Layer and pSHIELD Network Adapter

In section 3.3, the terms and definitions of Network Layer were defined, along with the specific implementation features, which constitute the targeting core networking capabilities of a pSHIELD system. Also, throughout section 6.5, the stepped procedure of defining a conceptual architecture is described. Based on and “modifying” OSI model pSHIELD adopts the four layered approach, thoroughly described previously. Abstracting the network layer component, from the general conceptual architectural Figure 50:

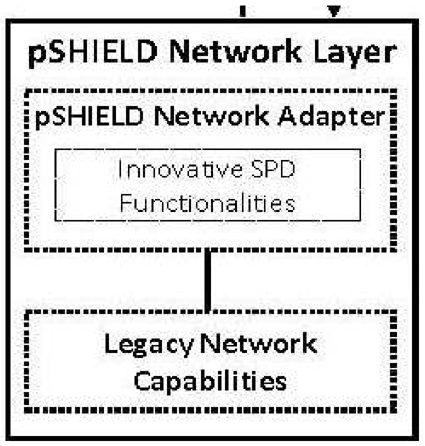


Figure 23 - pSHIELD Network Layer: Adapter and Legacy.

It can be deduced, that the pSHIELD Network Layer is constituted from a general sum of Legacy Network Services and the pSHIELD Network Adapter, responsible for adding innovative SPD functionalities in the system and being, practically, the main component implementing this Layer in the overall architecture.

Repeating from the reference architecture:

The ***pSHIELD Network Adapter*** includes a set of **Innovative SPD functionalities** interoperating with the legacy ESD network services (through the NS interface) and the pSHIELD Node Adapter (through the pS-NC interface) in order to enhance them with the pSHIELD Network layer SPD enabling technologies (such as **Smart Transmission**). This adapter is also in charge to provide (through the pS-NS interface) all the needed information to the pSHIELD Middleware adapter to enable the SPD composability of the Network layer legacy and Network pSHIELD-specific functionalities. Moreover, the pSHIELD Network Adapter translates the technology independent commands, configurations and decisions coming from the pS-NS interface into technology dependent ones and enforce them also to the legacy Network functionalities through the NS interface. The exact list of functionalities and services has to be determined in the process of this study.

The ***Legacy Network Services*** includes all the legacy network services (protocol stacks, routing, scheduling, Quality of Service, admission control, traffic shaping, etc.) provided by the Legacy Embedded

System Device which are not pSHIELD-compliant. In order to be pSHIELD compliant, these services should be enriched with pSHIELD SPD functionalities. This task is in charge of the pSHIELD Network Adapter. The exact list of functionalities and services has to be determined in the process of this study. The following tables contain an example of Legacy Network Services, regarding the protocol stack of two popular LAN networking technologies:

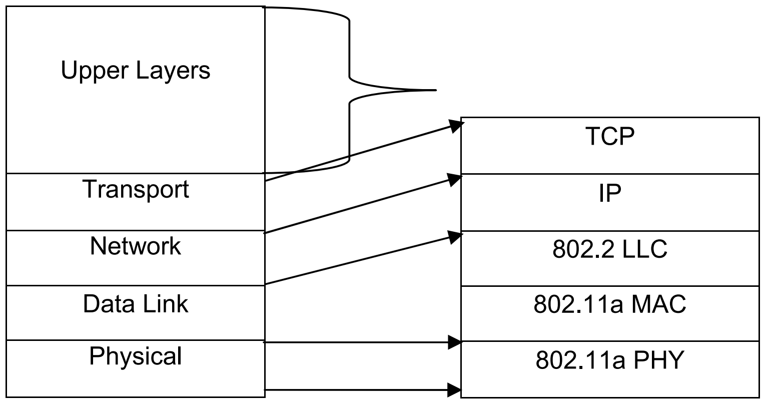


Table 4 - Legacy Network Stack (802.11a).

Layer	Protocol
Application	HTTP
Transport	TCP
Network	IP
Data Link	Ethernet
Physical	IEEE 802.3u

Table 5 - Legacy Network Stack (Ethernet).

The macroscopic overall layered view of pSHIELD system, is depicted below. The Network Layer's relative position and its interfacing points with other layers (mainly Middleware) are clearly shown. The basic element of ***pSHIELD Proxy*** (constituted by the Adapter and the Security Agent) hosts the SPD functionalities distributed in each of the four layers: (i) the pSHIELD Security Agent acts at Overlay; (ii) the pSHIELD Middleware Adapter acts at Middleware Layer; (iii) ***the pSHIELD Network Adapter acts at Network Layer*** and (iv) the pSHIELD Node Adapter acts at Node Layer. The logic is compositional, meaning that the Proxy could include *some* of the Adapters (not necessarily all) and inversely, a subset of the Adapters could form a Proxy, fitting specific application needs.

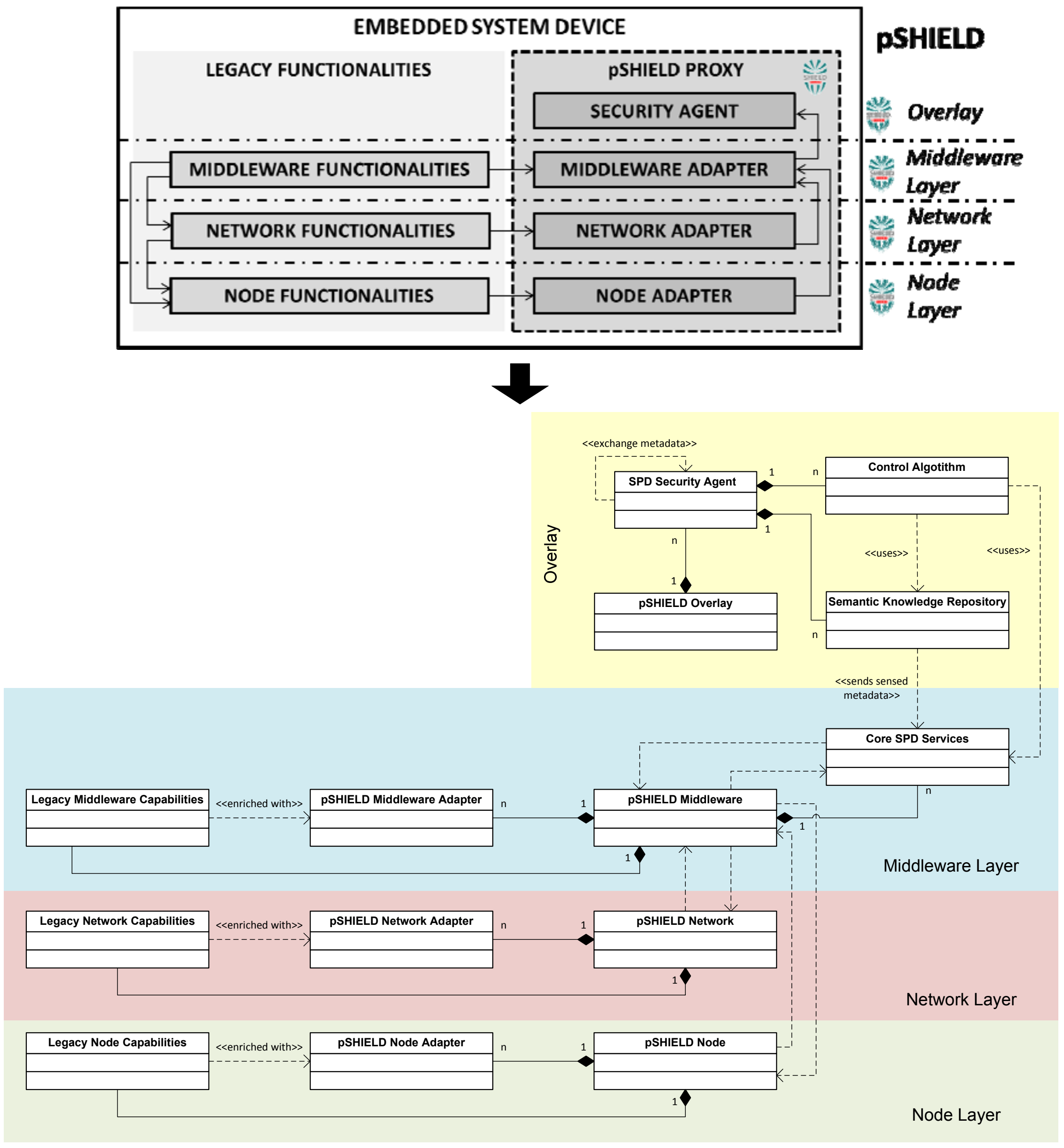


Figure 24 - ESD and pSHIELD System layered view.

6.2.2 Components and Devices

The actual nodes and their combinations in different types of sub-network or in a unified network, that will provide the users with the overall pSHIELD system solution, is the subject of this section. The components' selection leans on the monitoring of freight trains scenario, as sensing units of Intelligent Wagon concept implementation.

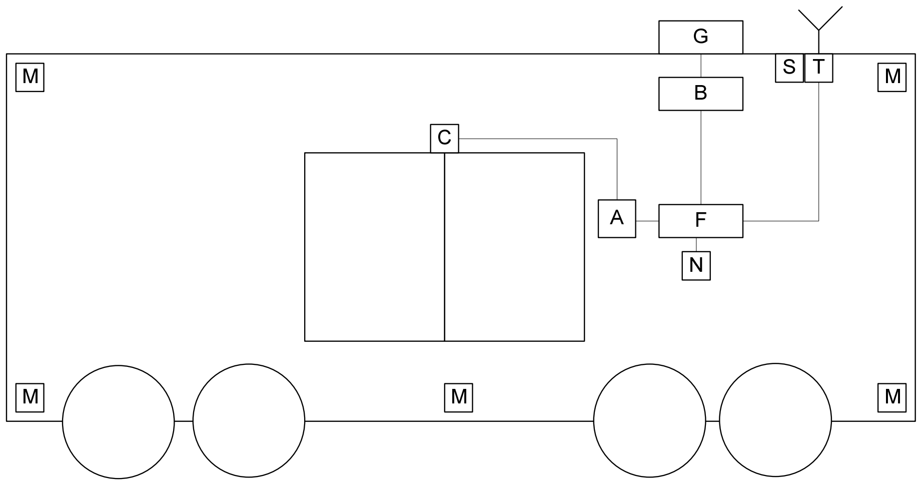


Figure 25 - Intelligent Wagon internal Network.

- A: Access control device powered on-demand (e.g. numeric keypad featuring “ON” button)
- B: Long lasting battery pack
- C: Magnetic contact or other very-low consumption intrusion detection device
- F: Central control unit featuring embedded CPU and OS
- G: Low size power generator (e.g. eolic, solar panel, piezoelectric pad) – optional
- M: Self-powered smart wireless sensor measuring vibrations, temperature, humidity, light, sound
- N: Gateway node for the wireless sensor network
- S: Satellite positioning antenna
- T: Wide area wireless transceiver (GSM, GPRS, UMTS, EDGE)

Regarding the implementation of the above scheme, three technologies have been, so far, foreseen:

- Micro Node – Sun SPOT Sensor Platform

They are platform integrated sensors. The processor board contains: temperature sensor, accelerometer sensor, light sensor. The supporting software is SUN SPOT Java Virtual Machine with OS functionality

- Personal Node - VIA Embedded Board

VIA EPIA N700 is a compact, low heat, power-efficient Nano-ITX board. The supporting software is Ubuntu Linux Kernel 2.6.32-24-generic and Java runtime environment (JRE) 1.6 for development

- M2M Platform – Telenor Shepherd Platform

A platform (named Shepherd) used for interoperability and integration that supports communication between connected devices (nano and micro nodes) and makes them accessible from anywhere at anytime. The supporting software, providing connectivity, is HTTP API, Connected Object Operating System (COOS)

The aim of Shepherd platform is to provide a highly usable and abstract presentation of services to users, networking the components in a way similar to the following:

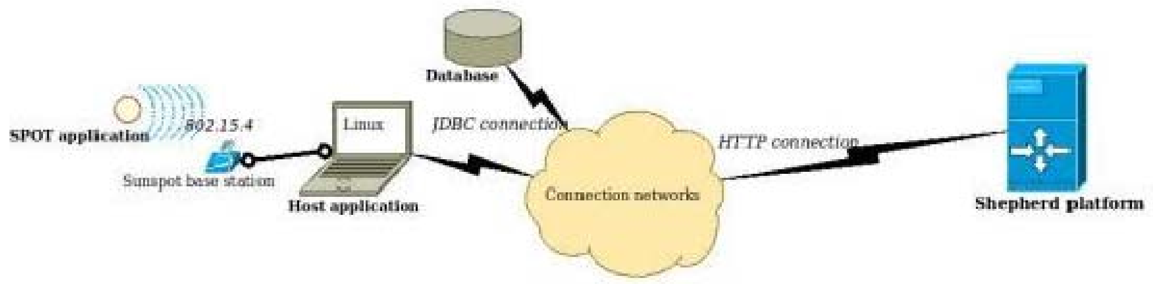


Figure 26 – Network of connected devices (M2M platform).

6.2.3 Functions and Services

6.2.3.1 Smart SPD Driven Transmission

Smart SPD information transmission is a feature of pSHIELD system, based on a Network Layer Service, usually called Software Defined Radio (SDR). An SDR platform will be used to provide smart transmission. It concerns a radio communication software system, implemented on embedded devices. It can receive and transmit a variety of different radio waveforms, based on the software used. It can, also, be integrated easily with hardware security modules. It allows to accommodate new standards and new Network Layer services as they emerge upgrading the terminal software without requiring to develop a new dedicated Embedded System Device.

The research activities on Network Layer functionalities of Embedded System Devices are focusing on the development of intelligent Cognitive Radio systems capable to understand and to be aware of the surrounding environment, allowing the exploitation of all the available wireless network services by using a single Embedded System Device. As an example, a Cognitive Radio (CR) could learn services available in a locally accessible wireless computer network and could interact with those networks by using its preferred protocols, so the users would not have confusion in finding the most suitable connection for, as an example, a video download or a printout. Additionally, a Cognitive Radio could select the carrier frequency and choose the transmitted waveforms according to the perceived environment and to reach a given goal, e.g. to avoid interference with existing wireless networks or to maximize the throughput while guaranteeing an acceptable Quality of Service (QoS).

The following picture shows the evolution of the radio network technologies for Embedded System Devices from traditional systems to Cognitive Radios.

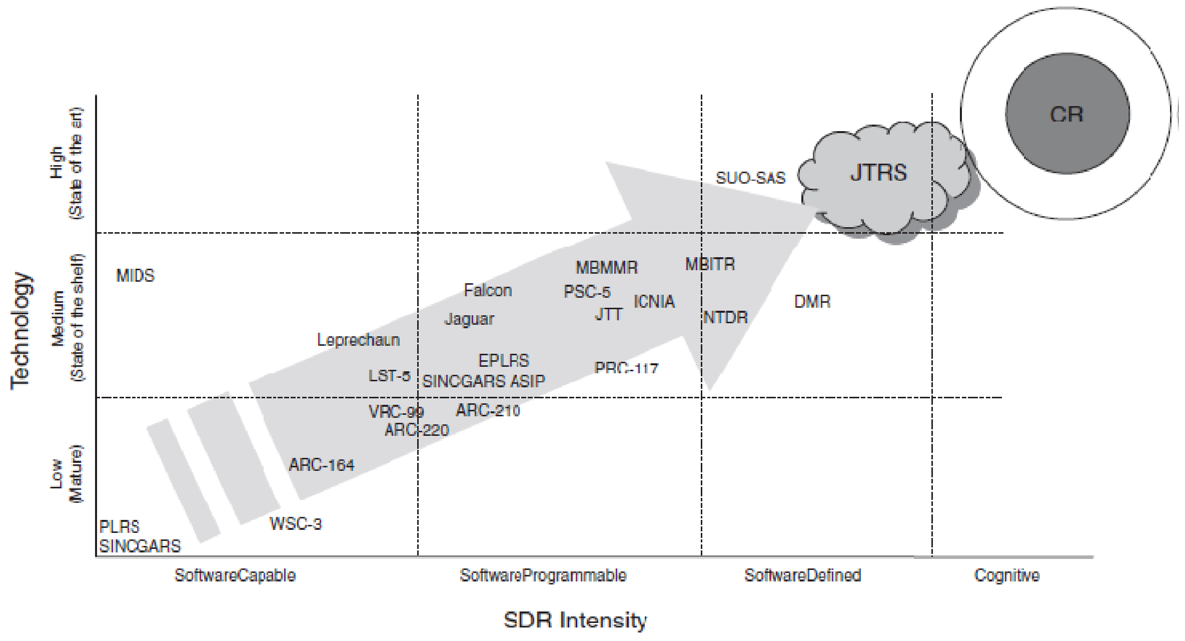


Figure 27 – SDR evolution.

6.2.3.2 Trusted and Dependable Connectivity

According to Haykin's definition[16], a Cognitive Radio is an intelligent wireless communication system that is aware of its surrounding environment (i.e., outside world), and uses the methodology of understanding-by-building to learn from the environment and to adapt its internal states to statistical variations in the incoming Radio-Frequency (RF) stimuli by making corresponding changes in certain operating parameters (e.g., transmit-power, carrier-frequency and modulation strategy) in real-time, with two primary objectives in mind: (i) highly Reliable and Dependable communications whenever and wherever needed and (ii) efficient utilization of the radio spectrum. As it is clear from this definition, the common keywords for an efficient Cognitive Radio are **Awareness** and **Reconfigurability**.

In a radio environment, Awareness means the capability of the Cognitive Radio to understand, learn, and predict what is happening in the radio spectrum, e.g., to identify the transmitted waveform, to localize the radio sources, etc. Reconfigurability is necessary to provide self-configuration of some internal parameters according to the observed radio spectrum. This is enormously important for both civilian and military applications especially when unforeseen situations happen and some Network Layers services are not available, guaranteeing trusted connectivity. It is now abundantly clear that the cognitive radios and their capabilities of dynamically maintaining a reliable and efficient communication can be significantly relevant in Security, Privacy and Dependability (SPD) driven applications where it is necessary to dynamically guarantee a high level of trustworthiness.

[16] Simon Haykin, "Cognitive Radio: Brain-Empowered Wireless Communications". IEEE Journal on Selected Areas in Communications, vol. 23, no. 2, February 2005

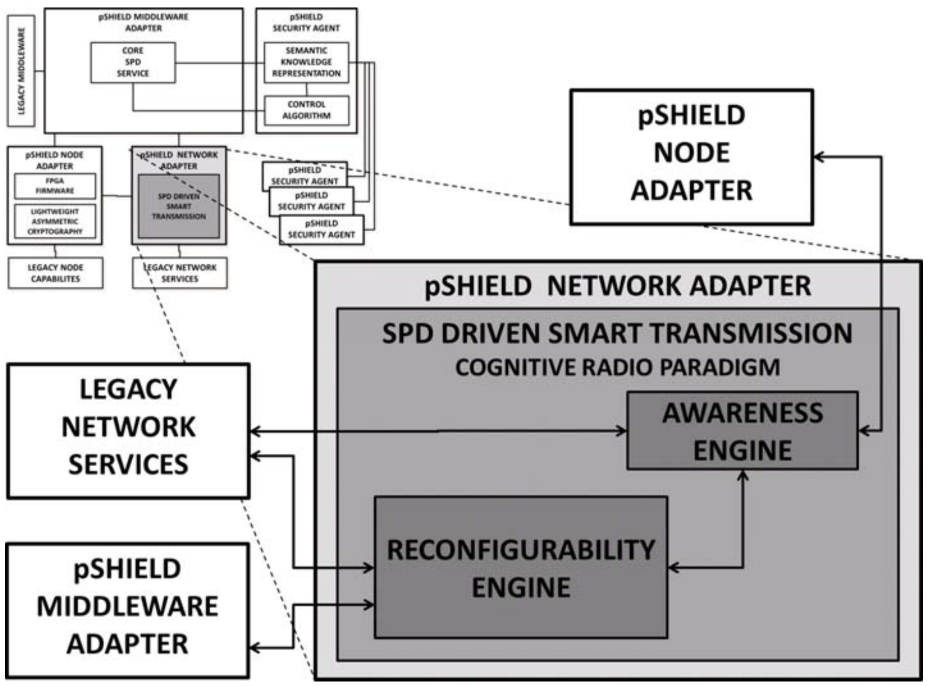


Figure 28 – pSHIELD NETWORK ADAPTER conceptual model.

As depicted in the above figure and according to the pSHIELD functional component architecture, the Smart SPD driven transmission is obtained applying a Cognitive Radio paradigm based on two main functional components: the Awareness Engine and the Reconfigurability Engine.

In particular, some Node Layer capabilities (e.g., antennas, cameras) and Network Layer services (e.g. available network resources, radio spectrum, number of active users, transmission protocols and standards, localization, etc.) can be used by the Awareness Engine to acquire a context awareness of the current radio environment. Then, some reasoning capabilities of the Cognitive Radio provided by the Reconfigurability Engine can be used to select the most appropriate configuration parameters useful to guarantee the needed SPD transmission.

It is important to note that, the Cognitive Radio capabilities are enormously attractive in a wide set of applications for both civilian (e.g., reliable communications, increased data-rate) and military (e.g., detect and decode enemy transmissions) scenarios. Although some encouraging preliminary results have been obtained in some practical environments, some open issues still remain and to obtain a general and multi-purposes cognitive radio is an open research problem. In general the main research challenges in this domain can be reduced to the following:

- Obtain a precise and concise representation of the radio environment (e.g., available resources, number of active users, transmission standard used, source localization) by using some Node Layer and Network Layer information
- Define the optimal configuration of the Network Layer according to a given goal (e.g., SPD metrics) and the perceived radio environment
- Develop algorithms and techniques for providing the capability of learning from the experience in order to face unforeseen and unexpected situations (e.g., a malicious user), that means the Embedded System Device's Network Layer is equipped with cognitive capabilities

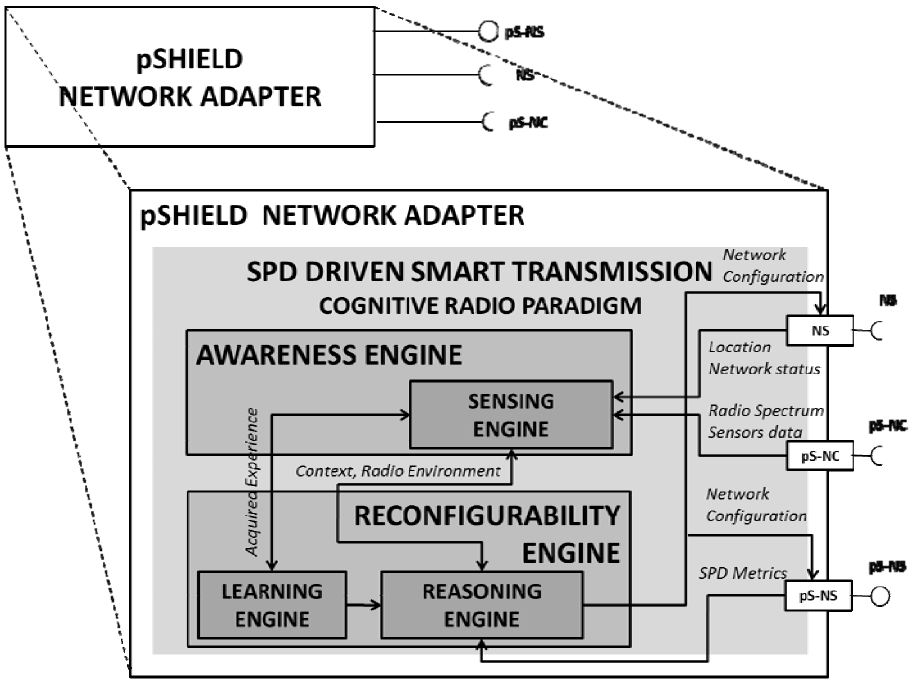


Figure 29 – pSHIELD Network Adapter with some examples of exchanged information.

In order to clarify how the main research challenges are mapped in the overall pSHIELD component architecture, a more detailed component architecture of the pSHIELD Network Adapter has been depicted in the above figure. One of the most important aims that a cognitive radio has to perform is to obtain an SPD communication whenever and wherever (e.g., in an unknown environment). This task is also known in the open literature as *Spectrum Sharing* and its main objective is to enhance the utilization of the radio spectrum, exploiting the unused resources, without causing harmful interference to existing active users in the monitored environment, while guaranteeing an acceptable level of Network Layer services.

To this end, the Awareness Engine has to gather information from the Legacy Network Services (through the NS interface) and from the pSHIELD Node Adapter (through the pS-NC interface), to acquire radio awareness (i.e. a precise and concise representation of the radio environment). For this purpose the Representation Engine collects all the needed information (radio spectrum, raw data from sensors, available network nodes and their status, etc.) and evaluates a precise and concise representation of the radio environment (i.e. the radio context). These algorithms and techniques which allow obtaining the radio awareness are known as *Spectrum Sensing*. For example, the Representation Engine can detect the active users in a monitored area, identify the used transmission standards, localize the radio sources, etc. This information is then provided to the Reconfigurability Engine to select a proper system Network Layer configuration according to a predefined goal of the Cognitive Radio and the surrounding environment.

To be compliant with the Cognitive Radio paradigm, the Awareness Engine should face even unknown radio contexts and react properly. In other words the Awareness Engine should be able to learn from the experience to face unforeseen and unexpected situations. This task is performed by the Learning Engine, which applies algorithms and techniques for providing cognitive capabilities (i.e. learning from the experience). The outcome of Learning Engine is a set of parameters that represent the acquired experience useful to configure optimally the Representation Engine and the Reconfigurability Engine.

The Reconfigurability Engine is in charge to identify the optimal configuration of the pSHIELD Network Layer according to a given goal (e.g., the SPD metrics provided by the pSHIELD Middleware), the perceived radio environment (provided by the Representation Engine) and the acquired experience (provided by the Learning Engine). As an example, given a specific radio context, the Reconfigurability Engine can be able to derive a new, optimal Network Layer configuration to establish a communication with an already active Embedded System Device by using its preferred transmission standard, carrier frequency and transmission power. This task is also known as *Opportunistic Communication* and allows identifying and exploiting an "opportunity" to establish a communication with the other players in a given context according to the surrounding environment conditions and the SPD goals.

6.2.3.3 Node Performance Network Services

Network services intended to increase SPD system level, through the effective administration and configuration of the nodes included, are listed below (the services concern mainly Power nodes and theis participation in LANs):

- Remote configuration
    - ✓ of BIOS and FPGA
- SNMP
    - ✓ A group of administrative computers manage the network devices
- Remote self-test and performance monitoring
- Privacy, authorization and authentication services

6.2.4 Communication protocols

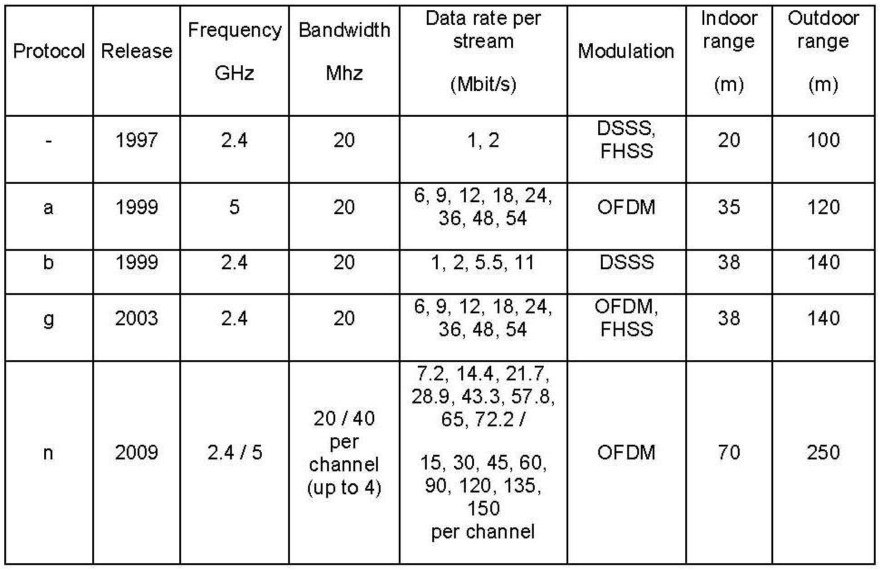
A reference to the communication technologies used by pSHIELD nodes is made at this point:

802.15.4

IEEE 802.15.4, the basis for ZigBee, is a standard which specifies the physical and MAC layers for Wireless Personal Area Networks (WPANs), designed to be rather frugal in its functional power demands. In pSHIELD the standard could serve the communication needs of small to medium range sensors and additionally of Legacy devices that potentially will complement a pSHIELD system environment. Micro node, for example, operates at 2.4 GHz based on 802.15.4 radio, with chip ChipCon TI CC2420. The RF transceiver offers compliant medium access control and physical interfaces for data rates up to 250 kbps. The 802.15.4 radio includes a DSSS (digital direct sequence spread spectrum) baseband modem providing a spreading gain (e.g. 9 dB in IRIS motes). The 2.4 GHz is divided into a number of channels, with a fixed width and spacing (e.g. the XM2110's Atmel radio can be tuned 16 channels from 11 (2.405 GHz) to 26 (2.480 GHz) each separated by 5 MHz.

IEEE 802.11

Alternatively most of the commercial sensors, legacy or belonging to pSHIELD hardware equipment, can have IEEE 802.11 as their radio interface. Being the basis of Wi-Fi (and colloquially known with this name), is probably the most popular family of standards for WLANs. Therefore in the framework of pSHIELD, seems as a candidate technology for the communication of Power node. It is IP based, issued in a set of amendments, the most known of which are summarized below:



Protocol	Release	Frequency GHz	Bandwidth Mhz	Data rate per stream (Mbit/s)	Modulation	Indoor range (m)	Outdoor range (m)
-	1997	2.4	20	1, 2	DSSS, FHSS	20	100
a	1999	5	20	6, 9, 12, 18, 24, 36, 48, 54	OFDM	35	120
b	1999	2.4	20	1, 2, 5.5, 11	DSSS	38	140
g	2003	2.4	20	6, 9, 12, 18, 24, 36, 48, 54	OFDM, FHSS	38	140
n	2009	2.4 / 5	20 / 40 per channel (up to 4)	7.2, 14.4, 21.7, 28.9, 43.3, 57.8, 65, 72.2 / 15, 30, 45, 60, 90, 120, 135, 150 per channel	OFDM	70	250

Table 6 - IEEE 802.11 family of standards.

Detailed description is probably out of scope, since plenty of references can be found in the bibliography. However a few synoptic network parameters are summarized here:

- Modulation
    - ✓ DSSS, FHSS, OFDM
- Medium Access
    - ✓ TDMA, FDMA, CDMA
- Keying Schemes
    - ✓ PSK, CCK, FSK
- Structured with Access Points and Hotspots
    - ✓ Basic Service Sets (BSSs) of 30 clients in 100m (theoretical maximum)
- Encryption
    - ✓ WEP, WPA (TKIP), WPA2 (CCMP)
- Authentication
    - ✓ 802.11i (WPA2)
    - ✓ EAP, EAP-TLS, EAP-TTLS, PEAP
- Frequency is divided into overlapping channels
    - ✓ e.g. Band: 2.4000–2.4835 GHz, channels: 13, width: 22 MHz, spaced: 5 MHz, channel 1: 2.412 GHz, channel 13: 2.472 GHz

Ethernet and Gigabit Ethernet

Another alternative in LAN communication, concerning Power node in our case, is Ethernet or IEEE 802.3 standard. It is a frame-based networking technique, defining a number of wiring standards for the Physical Layer and an addressing format and MAC procedures for the Data Link Layer. Some technical specifications, are shortly presented below (including the amendment of Gigabit Ethernet IEEE 802.3-2008):

- Medium access
  - ✓ CSMA/CD
- Data rate
  - ✓ Gigabit links may operate Full Duplex (Fast Ethernet Switch) or Half Duplex (Fast Ethernet Hub)
- Hardware Components
  - ✓ Hubs, Switches, Routers
  - ✓ Media converters (change twisted pair copper wires to fiber optic cabling)

6.2.5 Interfaces

During the incremental presentation of pSHIELD reference system architecture, we defined a generic interface NS (Network Services), responsible for Network Layer services provision. It is an integral element of all node types in the architecture hierarchy (L-ESD → pS-ESD → pS-SPD-ESD) and manages their Network Layer communication and functionalities. The innovative SPD functionalities implemented in pSHIELD introduced functional blocks and subsystems (e.g. Network Adapter) interoperate with Legacy systems through network interface NS.

There is also interface pS-NS, conveying all the needed information to the pSHIELD Middleware adapter, in order for the SPD composability of the Network layer legacy and Network pSHIELD-specific functionalities to be enabled.

A more detailed registration of all layer interfaces of pSHIELD, can be found in § 6.6.-6.7.

6.2.6 Security

6.2.6.1 Cryprographic Algorithms

According to the ISO/IEC 27002 information security standard, the objective of network security is the preservation of three principles:

- Confidentiality: the communication data are only disclosed to authorized subjects
- Integrity: the data in the communication retain their veracity and are not able to be modified by unauthorized subjects
- Availability: authorized subjects are granted timely access and sufficient bandwidth to access the data

In the selected application framework of monitoring hazardous material transported via railway, the necessity of a robust protection against malicious actions rises. Cryptography has been used for many years to provide security and information protection against different forms of attacks.

In previous section, during Node layer description (6.1), some cryptographic algorithms and mechanisms, possibly implemented on the nodes, were introduced. 802.15.4, the communication standard of the majority of pSHIELD nodes, includes symmetric cryptography, to protect data payload. Versions of Advanced Encryption Standard (AES) can be realized on the nodes. Trusted Platform Module (TPM) is another possible approach. However it should be noted that every algorithm appliance (accompanied by software or hardware extras) on the nodes comes with the trade-off of increasing constraints, such as cost, size, memory and energy efficiency and therefore has to be dealt cautiously.

6.2.6.2 IPsec

Security and appliance of corresponding protocol suites in wireless networks can be realized in many (if not all) layers of OSI model. A reference to Internet Protocol Security (IPsec), which implements both Encryption and Authentication in Network Layer, follows. IP should not be confused with Network Layer, in general, since the former is an implementation (the most popular though) of the latter. IPsec is the protocol suite for securing IP. It is an end-to-end scheme that can be used to protect sensible data transfer between hosts or gateways. In other words, IPsec is the “interconnecting” security scheme. Connecting with the known protocols, it is worth referring that IPsec overcomes RADIUS vulnerability concerning the latter’s lack of (per) packet authentication for access request packets. Originally designed for IPv6 version of Internet Protocol, IPsec is one of the commonest protocols for securing, through encryption, VPNs and in general remote accessing to private LANs. Another use concerns securing the path between Access Point and Authentication Server, during the authentication stepped procedure of a client’s request towards a wireless network. Often IPsec is adopted as a holistic solution for LANs and WLANs protection in the borders of an enterprise and there also lies its potential usefulness in pSHIELD network. By doing so, the network designer replaces Layer 3 (Network, e.g. IP) with *IPsec layer*, emphasizing users’ identity and credentials against mere IP addresses. The advantage of the approach rests in the fact that the notion of “security” acquires a more hardware based dimension. For example, routers would be necessary to be employed to route based not only in IP addresses but on other connection characteristics, also (IPsec related associations).

In the analysis of pSHIELD functional architecture, the notion and criticality of Security Agents was illustrated. Being part of the pSHIELD Proxy component, Security Agent is charged with the aggregation of information from the pSHIELD Middleware Services and from other Security Agents connected on the same Overlay, composing instances of pSHIELD subsystems to serve corresponding per demand needs. IPsec poses a strong candidacy as the network security mechanism foreseen for Security Agents. The overall security scheme would be again a composable one, maintaining the individual security protocols of each instance’s component, implementing simultaneously IPsec on top.

6.2.6.3 SSL

Along with IPsec, SSL, being the predecessor of TLS (Transport Layer Security), is the most popular cryptographic protocol, for securing communications across the Internet. It is used mostly to protect HTTP transactions, whereas other protocols concern IMAP (Internet Message Access Protocol) and POP3 (Post Office Protocol) and applications as web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP). SSL uses asymmetric cryptography to encrypt network data above OSI’s Transport Layer. It is composed of the following protocols:

1. Handshake protocol
2. Change Cipher Spec protocol

3. Alert protocol
4. Application Data protocol

6.3 Middleware Layer

6.3.1 Formalized conceptual model

The formalized conceptual model of the core SPD services has been conceived refining the pSHIELD middleware layer introduced in the previous section and has been derived from the study of the requirements of the pSHIELD application scenario (see deliverable D2.1.1 for more detail).

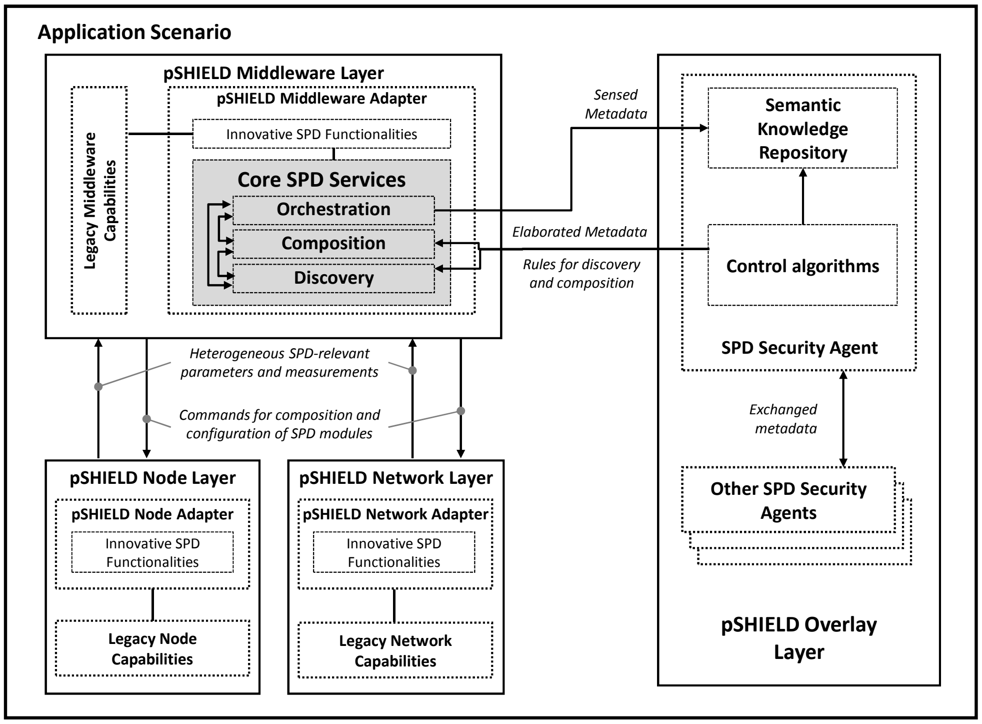

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Innovative SPD Functionalities
Core SPD Services
Orchestration
Composition
Discovery
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Exchanged metadata
Other SPD Security Agents
pSHIELD Overlay Layer
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities

Figure 30 – Core SPD services in the pSHIELD functional component architecture.

The Orchestration, Composition and Discovery functionalities are the enablers (i.e. the sensors and the actuators) 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.

It is worth to note that not all the core SPD services must be necessarily located in each pSHIELD Embedded System Device (pS-ESD). Indeed the pSHIELD component architecture depicted in above figure identifies the Discovery, Composition and Orchestration functionalities that must be supported by at least one pS-ESD in a network of Embedded System Devices. Moreover the core SPD services can be deployed applying centralized or distributed approaches. It is a matter of the precise application scenario to decide whether a specific functionality must be supported by each Embedded System Device (ESD). It is obvious that the more ESDs are equipped with the pSHIELD Middleware Adapter (resulting to be a pS-ESD), the more will be the coverage area and the effectiveness of the pSHIELD functionalities to guarantee a certifiable SPD level (based on common shared SPD metrics) over the whole system.

Let's see more in detail the formalized conceptual model of the Core SPD services, detailing the architecture depicted in previous figure and exploding the core SPD services into their functional components.

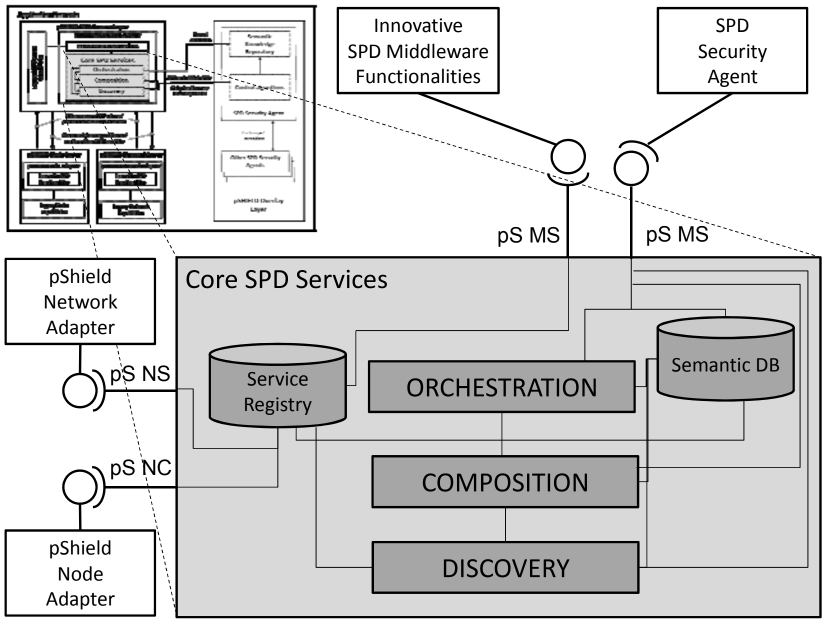


Figure 31 – Core SPD services in the pSHIELD functional component architecture.

Apart from the Discovery, Composition and Orchestration components already described in the previous section, the following additional conceptual entities have been introduced:

- *Service Registry*: it acts as a database to store the service entries. Any pSHIELD Node, Network or Middleware layer component can be registered here to be discovered
- *Semantic DB (Database)*: it holds any semantic information related to the pSHIELD components (interface, contract, SPD status, context, etc.). The use of common SPD metrics and of a shared ontology (derived from the formalized semantic model) to describe the different SPD aspects involved in guaranteeing a precise level of SPD, allows to dominate the intrinsic heterogeneity of the SPD components. Any semantic data is thus technology neutral and it is used to interface with the technology independent mechanisms applied by the pSHIELD Overlay

Focusing exclusively on the Core SPD services located in the pSHIELD Middleware Adapter, we can describe how it works when it is in an operative status. Let's consider a typical situation, where the whole system is properly working at runtime. The Orchestration functionality is in charge to monitor continuously the Semantic DB with the updated status of the functionalities operating at node, network and middleware layers. The pSHIELD Adapters are in charge to update in the Semantic DB their status.

Whenever the needed application SPD level changes and goes beyond the threshold, for any reason (e.g. due to external/internal unforeseen/ predictable events), the Orchestrator triggers the Overlay. The Overlay tries to react and to restore the SPD level back to an acceptable level identifying the best configuration rules. The Discovery and Composition are then triggered by the pSHIELD Overlay with the aim to apply the configuration rules. On the basis of the configuration rules, the Composition service makes use of the Discovery service to search for all the needed and available SPD components. The Composition service analyses the SPD components interfaces and contracts to determine which SPD components are required, which should be activated and in which order to make the configuration of SPD components properly work. Thus while the Overlay operates in a technology independent fashion, the Composition service operates all the needed low-level, technology-dependent activities to actuate the Overlay decisions.

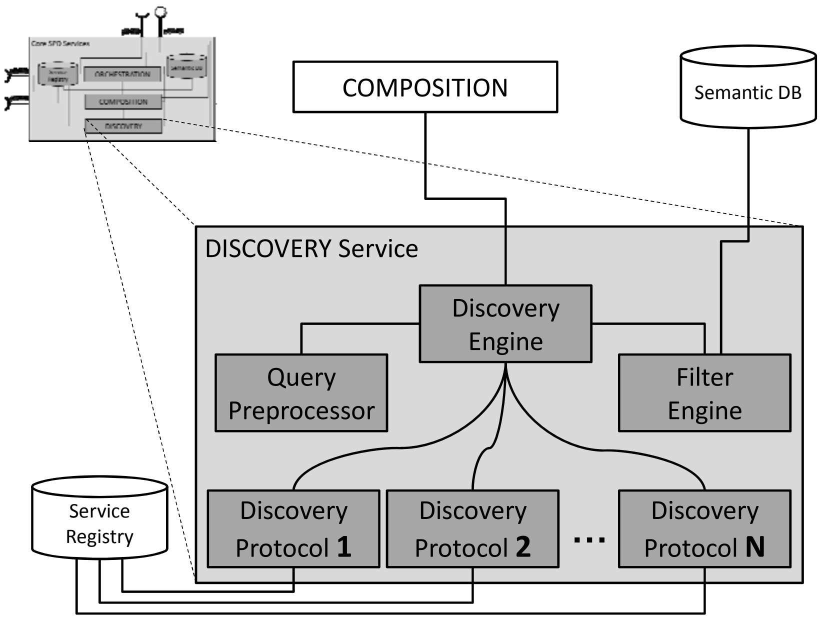


Figure 32 – Details of the Discovery core SPD service.

Zooming more in the detail the Discovery service, as shown in figure above, the following elements can be distinguished:

- **Discovery Engine**: is in charge to handle the queries to search for available pSHIELD components sent by the Composition service. The Discovery Engine manages the whole discovery process and activates the different functionalities of the Discovery service: (i) the query pre-processor to enrich semantically and contextually the query, (ii) the different discovery protocols to harvest over the interconnected systems all the available pSHIELD components, (iii) the Filter Engine to discard those components not matching with the enriched query
- **Query Pre-processor:** is in charge to enrich the query sent by the Composition service with semantic information related to the peculiar context. The query pre-processor can be configured by the Overlay to take care of the current environmental situation
- **Discovery Protocol:** is in charge to securely discover all the available SPD components description stored in the Service Registry, using a specific protocol (e.g. Service Location Protocol – SLP or Universal Plug and Play Simple Service Discovery Protocol – UPnP SSDP, etc.). Indeed the SPD component descriptions can be registered in different types of Service Registries, located everywhere in the network, using heterogeneous protocols to be inquired
- **Filter Engine:** 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, whose location is reported in the description of the SPD component

The composition engine tries to accomplish the pSHIELD Overlay configuration rules applying the following procedure:

1. Composition service triggers the Discovery service, sending a SPD component request, looking for those SPD components defined in the configuration rules provided by the Overlay
2. The Discovery Engine sends the request to the Query Preprocessor
3. The Query Preprocessor enriches the service request with contextual information and sends it back to the Discovery Engine
4. The Discovery Engine applies a global service discovery using heterogeneous Discovery Protocols, in order to collect as much available SPD functionalities as possible over the networked Embedded System Devices
5. Each Discovery Protocol interacts with the Service Registries reachable in the network and retrieves the SPD components' descriptions and provides them back to the Discovery Engine
6. The Discovery Engine collects the discovered descriptions and sends them to the Filter Engine
7. The Filter Engine applies a semantic filtering, retrieving the semantic metadata from the semantic DB, accordingly with the references reported in each SPD component description. The filtered list of component is then sent back to the Discovery Engine
8. The Discovery Engine sends the list of available, filtered SPD components to the Composition service
9. If the Composition service, considering the available SPD components is able to provide a new configuration, these components are activated, otherwise the Composition service advise the Overlay that it is not possible to apply its decision

It is important to note that the validity of this conceptual framework model is independent from the specific application scenario. On the basis of this conceptual framework it is possible to derive a number of possible alternative implementations, belonging to different pSHIELD compliant technology providers. If the interfaces and the operation between the different elements are respected, it is possible to setup heterogeneous systems with the enhanced pSHIELD SPD functionalities.

6.4 Overlay Layer

The following figure highlights in light red, some key functionalities and interfaces involving the Overlay layer.

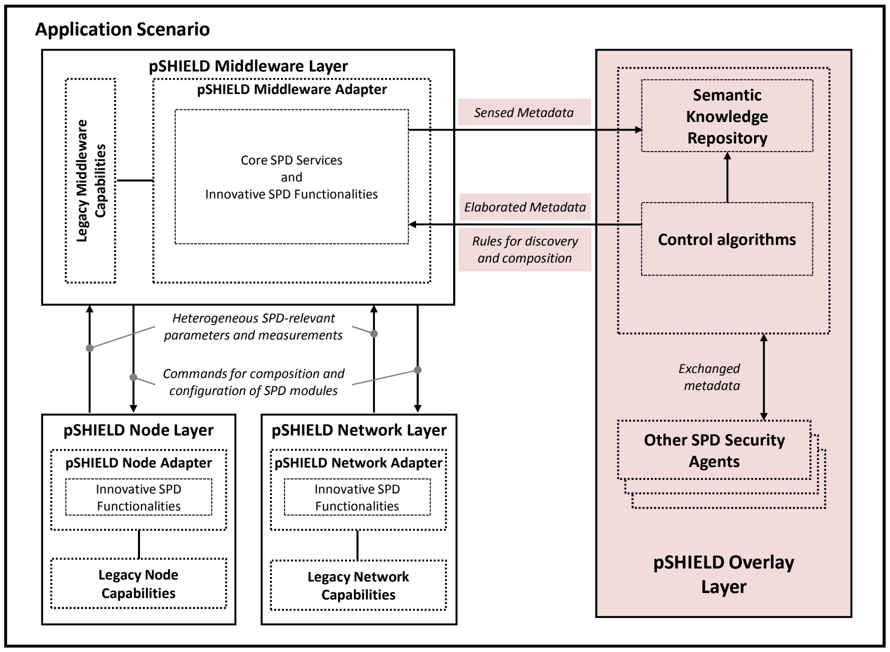

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 33 – pSHIELD overlay: a functional view

The Overlay consists of a set of SPD Security Agents, each one controlling a given pSHIELD subsystem. Subsystem identification has to be carefully performed scenario by scenario. Expandability of such a framework is obtained by enabling communication between SPD Security Agents controlling different sub-systems. As a matter of fact, the presence of more than one SPD Security Agent is justified by the need of solving scalability issues in the scope of system-of-systems: exponential growth of complexity can be overcome only by adopting the policy of *divide et impera* (divide and rule).

Each SPD Security Agent, in order to perform its work, exchange carefully selected information with the other SPD Security Agents, as well as with the three horizontal layers (node, network and middleware) of the controlled pSHIELD subsystem.

Each SPD Security Agent collects properly selected heterogeneous SPD-relevant measurements and parameters coming from node, network and middleware layers of the controlled pSHIELD subsystem; this information is used as valuable input for the Control Algorithms. Since the SPD Security Agent is mainly a software functionality and not a physical system itself, it needs the mediation of the pSHIELD Middleware Core SPD Services, as it has been explained in the previous section.

The heterogeneous data collected from the three horizontal layers are abstracted, translated into technology-independent metadata, semantically enriched and aggregated. The resulting metadata (referred to as *sensed metadata*) are stored in the Semantic Knowledge Representation of the considered SPD Security Agent. By so doing, this database stores a dynamic, semantic representation of the controlled pSHIELD subsystem.

In the considered SPD Security Agent, the representation mentioned above is used as a valuable, rich input for a set of intelligent, technology-independent, closed-loop Control Algorithms. These last, by using (as input) the above-mentioned representation and by adopting appropriate advanced methodologies able to profitably exploit such input, produce (as output) decisions aiming at guaranteeing, whenever it is possible, target SPD levels over the controlled pSHIELD subsystem. In the Composability feature described in the previous section, these decisions consist in a set of rules for discovery, configuration and composition of SPD components.

The decisions mentioned above are eventually actuated in the pSHIELD subsystem controlled by the considered SPD Security Agent, by exploiting the mediation of the middleware as detailed in the previous section.

Summarizing, each SPD Security Agent consists of two key elements:

i. the Semantic Knowledge Repository (i.e. a database) storing the dynamic, semantic, enriched, ontological aggregated representation of the SPD functionalities of the pSHIELD subsystem controlled by the SPD Security Agent

ii. the Control Algorithms generating, on the grounds of the above representation, key SPD-relevant decisions (consisting, as far as the Composability feature is concerned, in a set of discovery, configuration and composition rules)

The formalized conceptual model of a pSHIELD Security Agent, described in the previous section, is reported in the below figure.

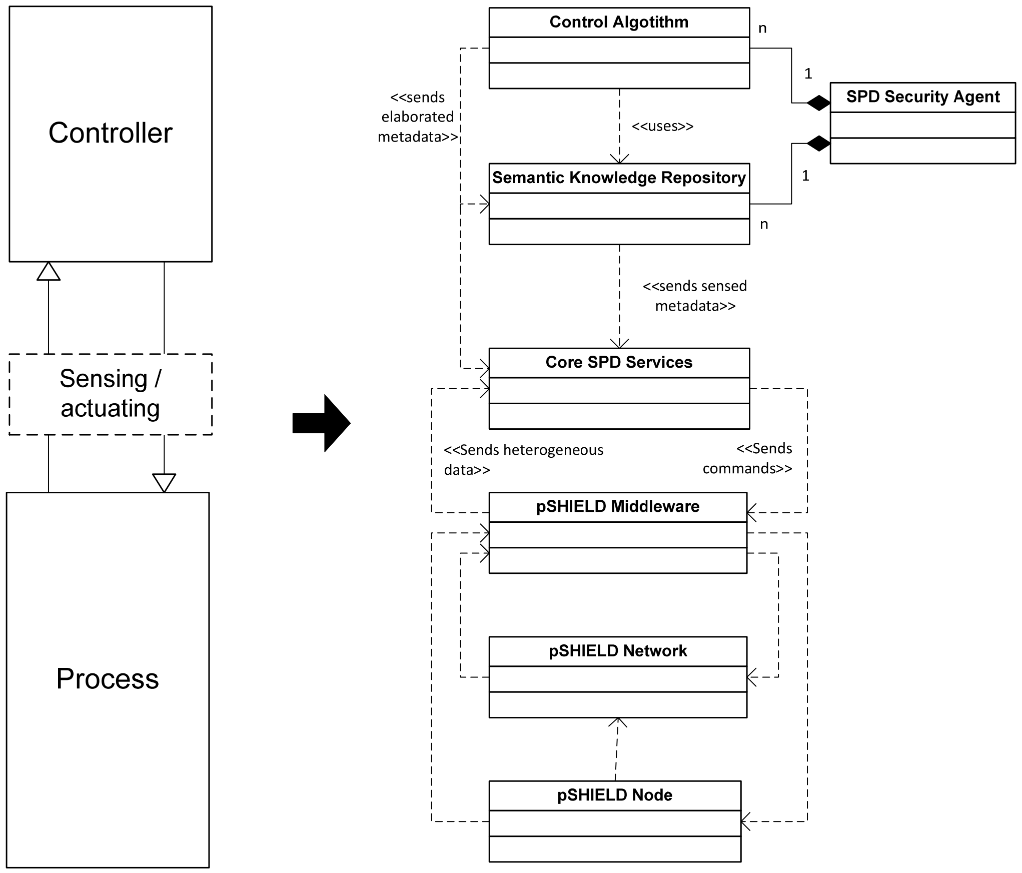


Figure 34 – Formalized conceptual model for pSHIELD Security Agent.

The figure also highlights the correspondence between the class diagram and a classical feedback control scheme (including Process, Controller and Sensing/Actuation functionalities) where:

- the Process to be controlled is represented by the three horizontal layers (Node, Network and Middleware)
- the Controller is the Security Agent supported by the Semantic Knowledge Repository
- Sensors/Actuators are represented by the Core SPD Services lying at the pSHIELD Middleware layer

The formalized conceptual model of the pSHIELD Overlay is shown in the following figure. Taking into account that the pSHIELD Overlay can include several Security Agents each one controlling a pSHIELD subsystem, the pSHIELD Overlay model is easily obtained by composing in parallel the feedback control scheme relevant to each Security Agent presented in the previous figure. The coordination and information exchange between the different controllers (i.e. SPD Security Agents) is performed by means of metadata (referred to as *exchanged metadata*). The advantages in terms of flexibility entailed by this modular approach are evident.

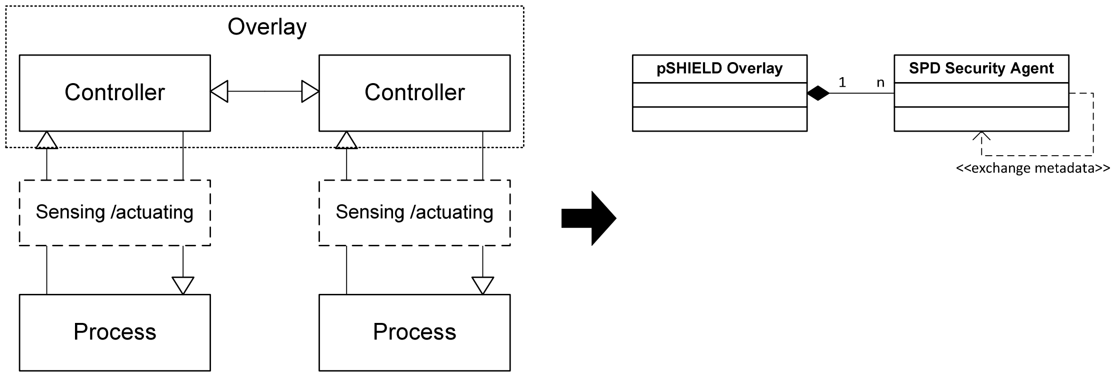


Figure 35 – Formalized conceptual model for the pSHIELD Overlay.

6.5 System Overall Architecture

In order to formally describe the pSHIELD system overall architecture, we decided to use the UML component diagram formalism, where there could be identified the following formal elements:

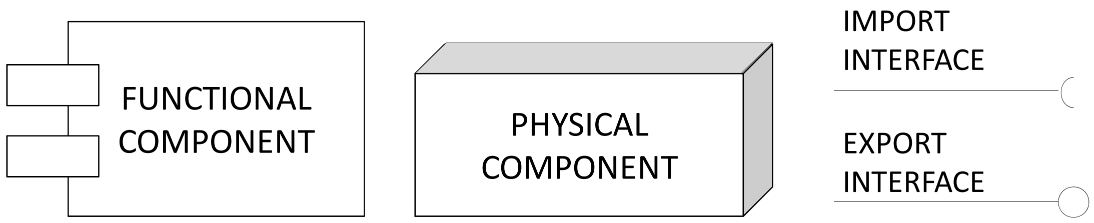


Figure 36 – pSHIELD functional architecture formalism.

A *functional component* describes a functional entity that, in general, does not have necessarily a physical counterpart (e.g. a software functionality, a middleware service, an abstract object, etc.). A *physical component* describes an entity that can be mapped into a physical object (e.g. a hardware component). A functional component, as well as a physical component, can require or provide functionalities, properties, connections, services, configuration parameters, energy, etc. To model these aspects, we use the *interface* symbol. An interface can be used to import all the needed elements to make a component to properly work. An interface can also be used to export some elements as outcome of the component operations or physical structure.

From the pSHIELD perspective, there exist three different types of Embedded System Devices (ESDs):

Legacy Embedded System Device (L-ESD): it represents an individual, atomic physical Embedded System device characterized by legacy Node, Network and Middleware functionalities. This device can be modelled as depicted in the following figure. So, the legacy functionalities of an L-ESD can be partitioned into three subsets:

- Node layer functionalities: hardware functionalities such as processors, memory, battery, I/O interfaces, peripherals, etc.
- Network layer functionalities: communication functionalities such as connectivity, protocol stacks, etc.
- Middleware layer functionalities: firmware and software functionalities such as services, functionalities, interfaces, protocols, etc.

The L-ESD exposes three interfaces: (i) the legacy, technology-dependent middleware services, (ii) the legacy, technology-dependent network services and (iii) the legacy, technology-dependent node capabilities.

pSHIELD Embedded System Device (pS-ESD): is a L-ESD equipped at least with the minimal set of pSHIELD functionalities at Middleware Layer. This device can be modelled as depicted in the second of the following figures: The pS-ESD exposes the same functionalities as the L-ESD plus an additional interface: the pSHIELD Middleware layer services.

pSHIELD SPD Embedded System Device (pS-SPD-ESD): is a pS-ESD equipped at least with the minimal set of pSHIELD Overlay functionalities. This device can be modelled as depicted in the third of the following figures.

The pS-SPD-ESD exposes the same functionalities as the pS-ESD plus an additional interface: the pSHIELD Overlay layer SPD services provided by a so-called *Service Agent* operating in that ESD.

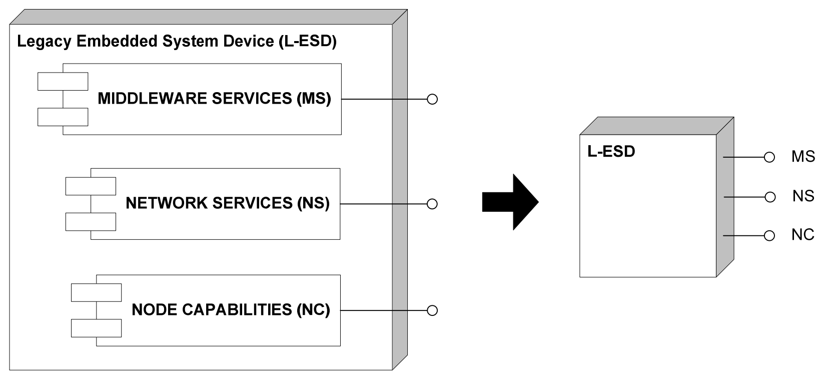


Figure 37 – Legacy Embedded System Device (L-ESD) with its exposed functionalities.

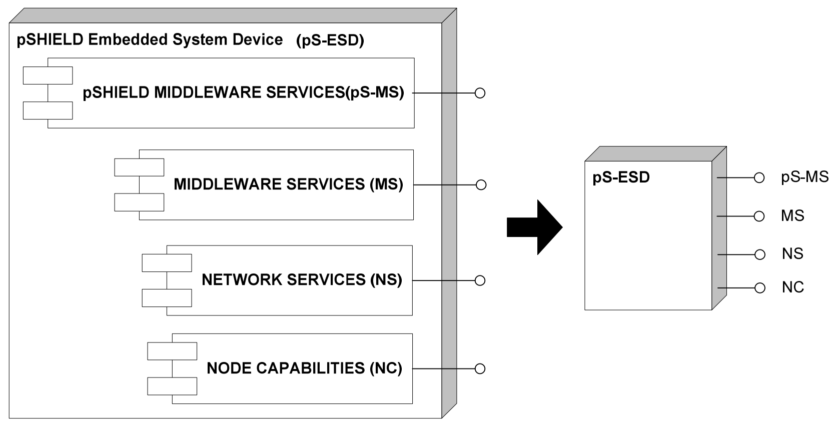


Figure 38 – pSHIELD Embedded System Device (pS-ESD) with its exposed functionalities.

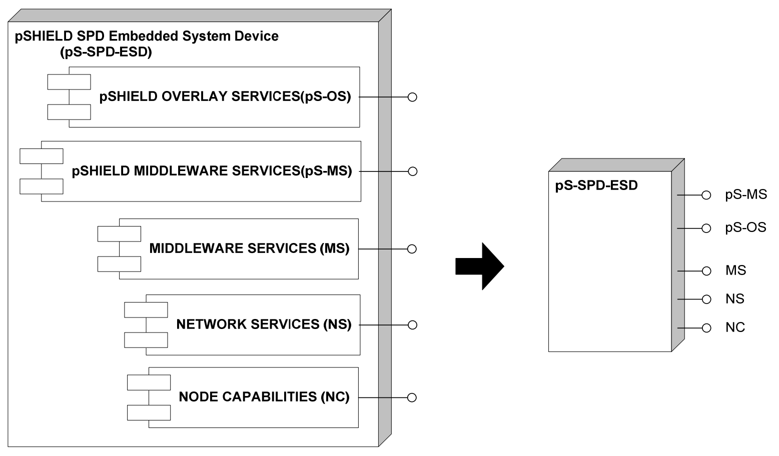

pSHIELD SPD Embedded System Device (pS-SPD-ESD)
pSHIELD OVERLAY SERVICES(pS-OS)
pSHIELD MIDDLEWARE SERVICES(pS-MS)
MIDDLEWARE SERVICES (MS)
NETWORK SERVICES (NS)
NODE CAPABILITIES (NC)
pS-SPD-ESD
pS-MS
pS-OS
MS
NS
NC

Figure 39 – pSHIELD SPD Embedded System Device (pS-SPD-ESD).

We can define the architecture of a pSHIELD Subsystem (pS-S) as a set of Embedded System Devices including several L-ESD, connected to several pS-ESD and one and only one pS-SPD-ESD. Connections between two generic ESDs (L-ESD, pS-ESD or pS-SPD-ESD) can be performed, by means of legacy functionalities at Node, Network and/or Middleware layer, through the so-called NC, NS and MS functionalities, respectively. This means, for example, that the legacy Node layer capabilities (e.g. legacy physical connectors) of an L-ESD can be used to connect it to a pS-SPD-ESD having a compatible Node layer capability (e.g. the same connector format factor). The connection of a pS-ESD with another pS-ESD or with a pS-SPD-ESD can also be performed by exploiting the additional pSHIELD Middleware layer functionalities exposed by the pS-MS interface. Each pSHIELD Subsystem (pS-S) must have one and only one pS-SPD-ESD, so that a pS-SPD-ESD can be connected to another pS-SPD-ESD by means of so called pS-OS functionalities. A pSHIELD Subsystem is depicted in the following figure.

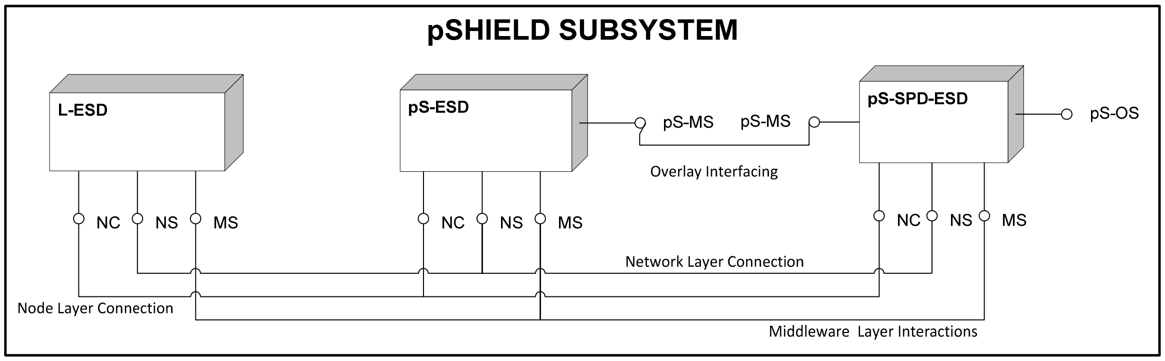


Figure 40 – pSHIELD Subsystem architecture decomposed into ESD components

Let's consider a pSHIELD system including several Embedded System Devices belonging to the three above defined types of ESD: L-ESD, pS-ESD and pS-SPD-ESD. The pSHIELD system architecture can be represented as a set of pSHIELD subsystems each connected to the other by means of the overlay interfaces provided by the Service Agent:

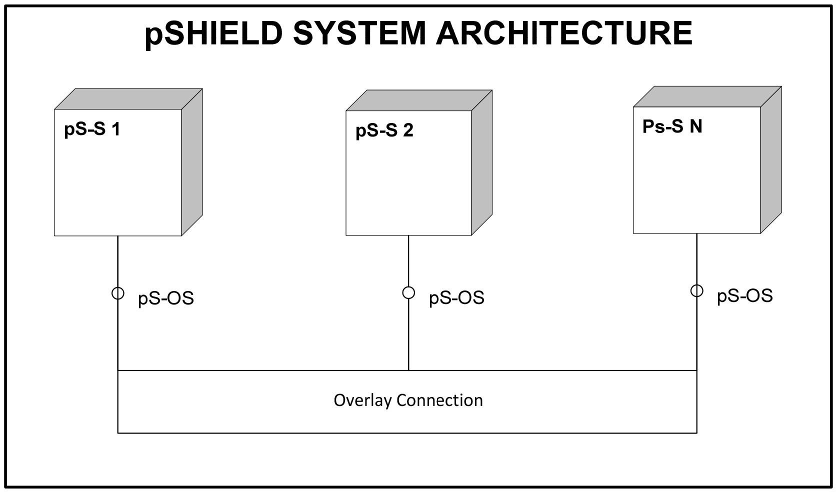


Figure 41 – pSHIELD System Architecture decomposed into pSHIELD Subsystem

The figure below shows the pSHIELD System Architecture highlighting the various ESD types.

In particular, for the sake of clarity, only the pSHIELD Subsystem 1 has been exploded to explicit the various ESD types. Specifically, within the pSHIELD Subsystem 1, (i) the connections among the pS-ESD and the L-ESDs have been grouped in the “Legacy Middleware Network Node” cloud, (ii) the connections among the pS-ESDs and the pS-SPD-ESD have been grouped in the “pSHIELD Middleware” cloud, (iii) the connections among different pSHIELD Subsystem (i.e. between different pS-SPD-ESDs) have been grouped in the “Overlay” cloud.

The Overlay consists of a set of SPD Security Agents located each in a different pS-SPD-ESD. Each SPD Security Agent controls a different pSHIELD subsystem. Subsystem identification has to be carefully performed scenario by scenario. Expandability is obtained by enabling communication between SPD Security Agents taking care of different pSHIELD sub-systems.

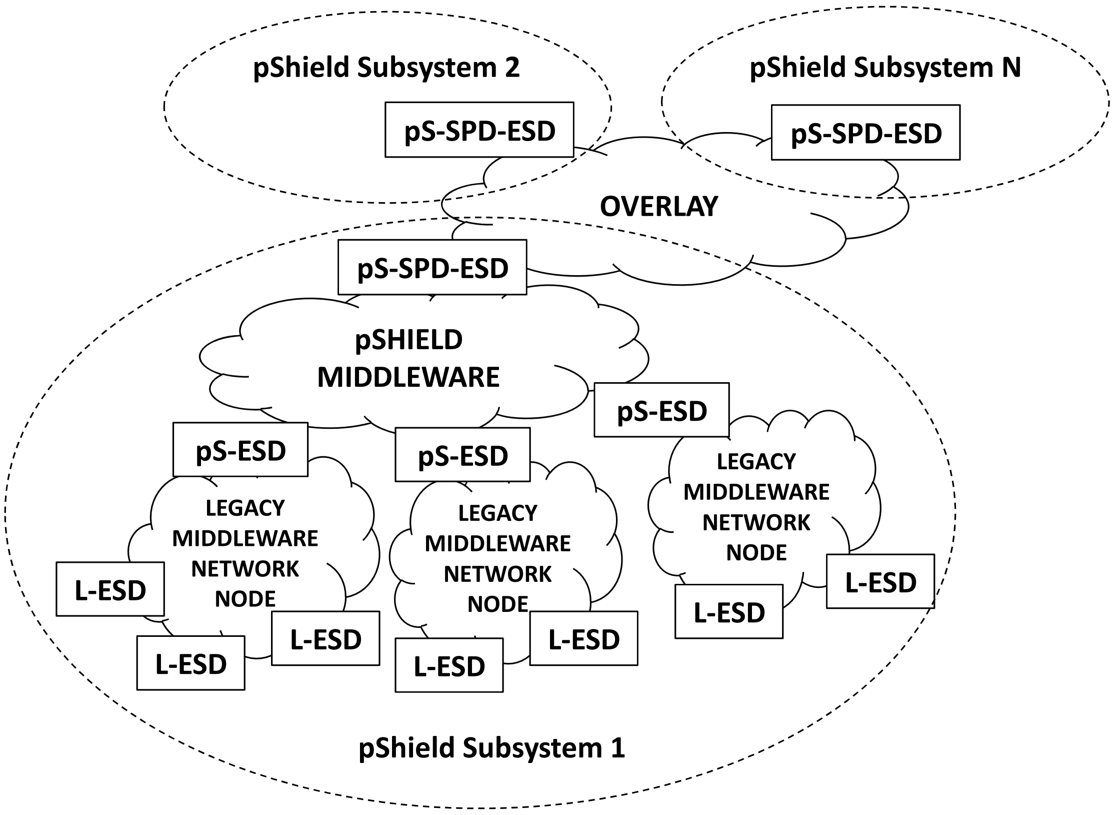


Figure 42 – pSHIELD System Architecture highlighting the ESD types.

In order to introduce the next sections, we should refine the ESD model. Exploding the model of a pS-SPD-ESD, it is possible to identify that its external functionalities are provided by the interaction of two components: the L-ESD and the pSHIELD Proxy. While we have already defined the L-ESD component, the pSHIELD Proxy is a new component.

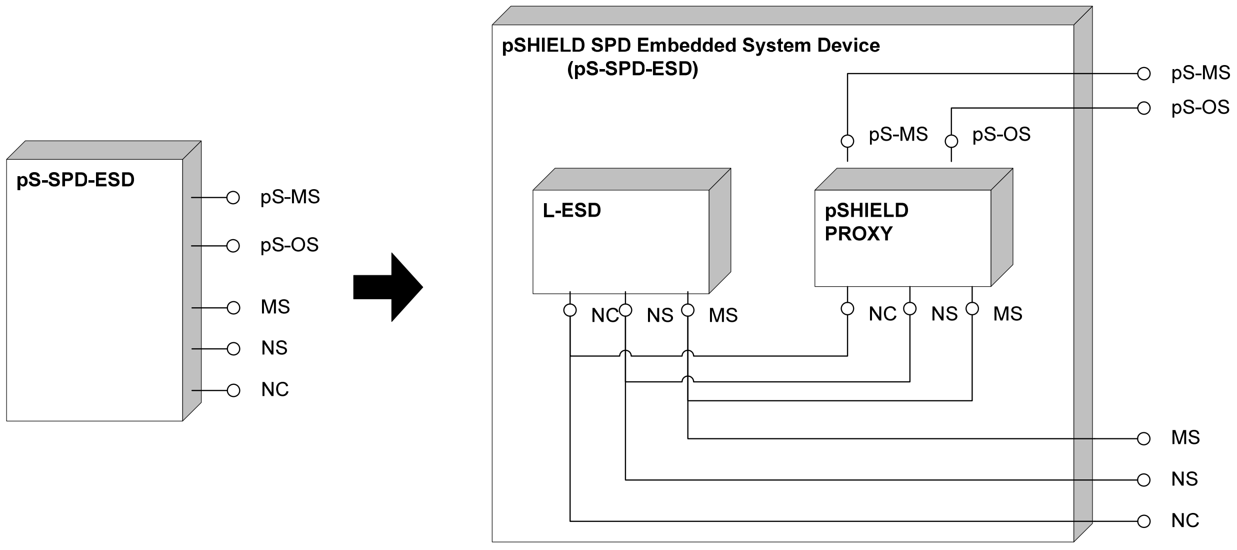


Figure 43 – pS-SPD-ESD architecture.

A ***pSHIELD Proxy*** (pS-P) is a technology dependent component of a pS-SPD-ESD that, interacting with the available legacy Node, Network and Middleware capabilities and functionalities (through the NC, NS and MS interfaces, respectively), provides all the needed pSHIELD enhanced SPD functionalities. From the above figure, it is clear that the pS-SPD-ESD functionalities belongs both to the L-ESD component (MS, NS and NC interfaces), as well as to the pSHIELD Proxy component (pS-MS and pS-OS interfaces).

The pSHIELD Proxy component can be further decomposed in two components (see following figure), namely the pSHIELD Adapter and the Security Agent. The rationale behind these two components, as well as their architecture, is hereinafter explained.

The ***pSHIELD Adapter*** is a technology dependent component in charge of interfacing with the legacy Node, Network and Middleware functionalities (through the MS, NS and NC interfaces). The legacy functionalities can be enhanced by the pSHIELD Adapter in order to make them *pSHIELD-compliant,* i.e. they become SDP *legacy device components*, which can be composed by other SPD components, according to the SPD Composability approach. In addition, the pSHIELD Adapter includes *Innovative SPD functionalities* which are SPD *pSHIELD-specific components,* which can be composed by other SPD components. The pSHIELD Adapter exposes the technology independent pSHIELD Middleware layer functionalities that are used by the Security Agent component.

The ***Security Agent*** is a technology-independent component in charge of aggregating the information coming from the pSHIELD Middleware Services provided by the internal pSHIELD Adapter or by other pSHIELD Proxies located in the same subsystem. The Security Agent is also in charge of gathering the information coming from other Security Agents connected on the same Overlay (through the pS-OS interface). The Security Agent includes proper control algorithms working on the basis of the available information; the decisions taken by these Control Algorithms are enforced through the pS-MS and the pS--OS interfaces.

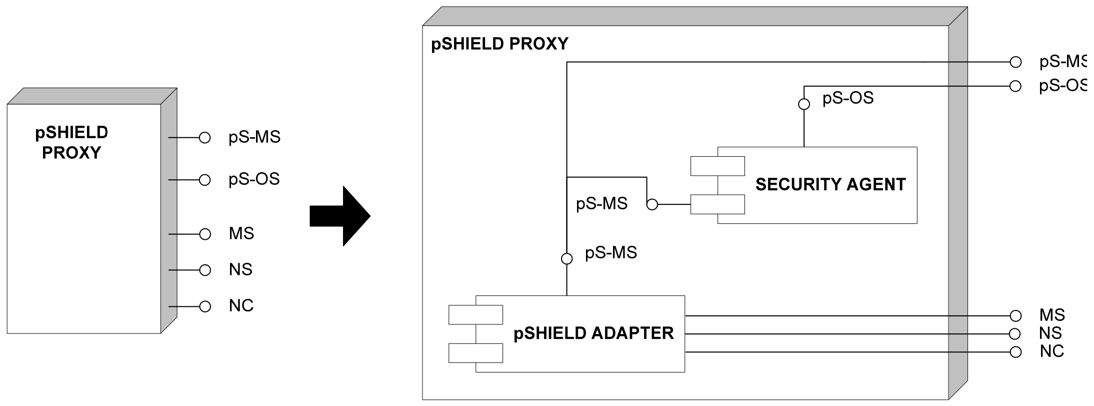


Figure 44 – pSHIELD Proxy component architecture.

The Security Agent component can be further decomposed in two components (see following figure), namely the Semantic Knowledge Representation and the Control Algorithms. The rationale behind these two components, as well as their architecture, is hereinafter explained.

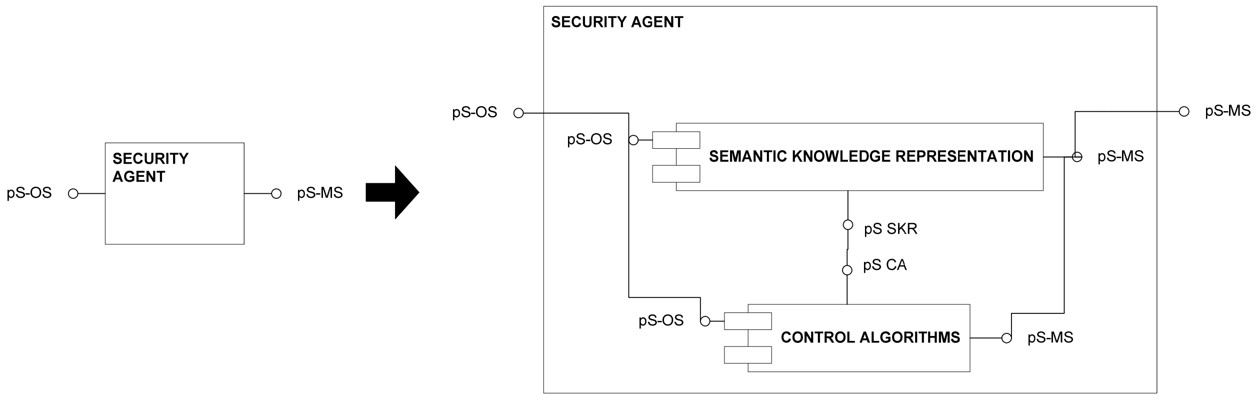


Figure 45 – pSHIELD Security Agent component architecture

The ***Semantic Knowledge Representation*** is in charge of bi-directionally exchanging technology-independent (and semantic enriched) information from the pS-MS and the pS-OS interfaces. It is also in charge to provide such information via the pS-SKR interface to the Control Algorithms component.

The ***Control Algorithms*** retrieves the aggregated information on the current SPD status of the subsystem, as well as of the other interconnected subsystems, by the pS-CA interface connected to the Semantic Knowledge Representation; such retrieved information is used as input for the Control Algorithms. The outputs of the Control Algorithms consist in decisions to be enforced in the various ESDs included in the pSHIELD subsystem controlled by the Security Agent in question; these decisions are sent back via the pS-MS interface, as well as communicated to the other Security Agents on the Overlay, through the pS-OS interface.

The pSHIELD Adapter can be further decomposed as in three components (see following figure), namely the pSHIELD Node Adapter, the pSHIELD Network Adapter and the pSHIELD Middleware Adapter. The rationale behind these three components, as well as their architecture, is hereinafter explained.

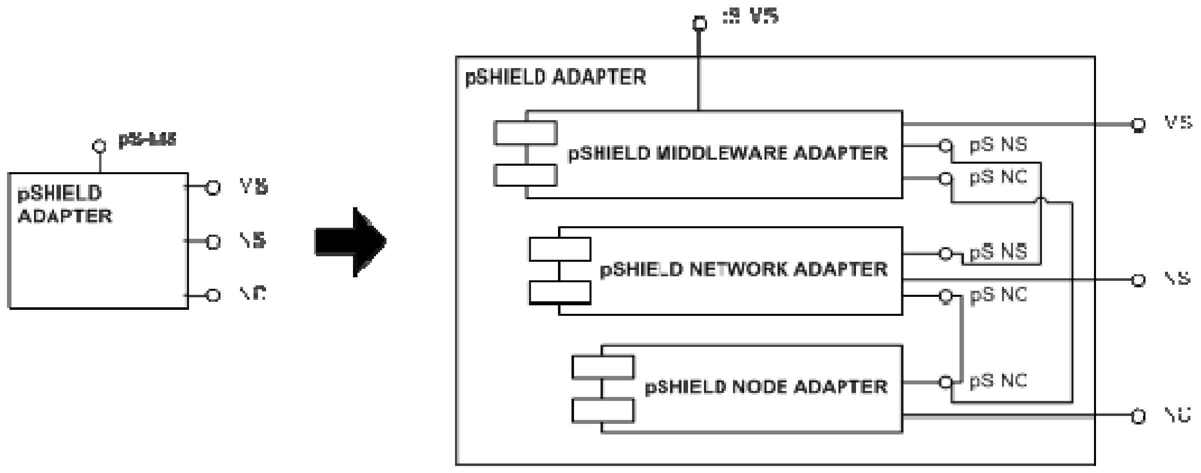


Figure 46 – pSHIELD Adapter component architecture

- The ***pSHIELD Node Adapter*** includes a set of **Innovative SPD functionalities** interoperating with the legacy ESD node capabilities (using the NC interface) in order to enhance them with the pSHIELD Node layer SPD enabling technologies (such as **FPGA Firmware** and **Lightweight Asymmetric Cryptography**). This adapter is in charge to provide (through the pS-NC interface) all the needed information to the pSHIELD Middleware adapter to enable the SPD composability of the Node layer legacy and Node pSHIELD-specific functionalities. Moreover, the pSHIELD Node Adapter translates the technology independent commands, configurations and decisions coming from the pS-NC interface into technology dependent ones and enforce them also to the legacy Node functionalities through the NC interface.
- The ***pSHIELD Network Adapter*** includes a set of **Innovative SPD functionalities** interoperating with the legacy ESD network services (through the NS interface) and the pSHIELD Node Adapter (through the pS-NC interface) in order to enhance them with the pSHIELD Network layer SPD enabling technologies (such as **Smart Transmission**). This adapter is also in charge to provide (through the pS-NS interface) all the needed information to the pSHIELD Middleware adapter to enable the SPD composability of the Network layer legacy and Network pSHIELD-specific functionalities. Moreover, the pSHIELD Network Adapter translates the technology independent commands, configurations and decisions coming from the pS-NS interface into technology dependent ones and enforce them also to the legacy Network functionalities through the NS interface.
- The ***pSHIELD Middleware Adapter*** which can be further partitioned in the **Core SPD services** and in the **Innovative SPD functionalities.** These two components are linked through the pS-MS interface. The figure below shows the pSHIELD Middleware Adapter component architecture.
- The *Core SPD Services* are in charge to discover all the available functionalities at Node, Network and Middleware layers and to describe them in a technology-independent fashion. All the information is sent to the Overlay through the pS-MS interface. The pSHIELD Middleware Adapter should also carry into operation the decisions taken by the Overlay and communicated via the pS-MS interface by actually composing the discovered SPD functionalities. The pSHIELD Middleware Adapter includes a set of **Innovative SPD functionalities** interoperating with the legacy ESD middleware services (through the MS interface) in order to make them discoverable and composable SPD functionalities.

The second of the following figures shows the pS-ESD architecture. As expected, a pS-ESD is an enhanced L-ESD with some additional pSHIELD capabilities: the ones provided by the pSHIELD Adapter component. However, the pS-ESD is not equipped with a Security Agent (otherwise it would have become a pS-SPD-ESD).

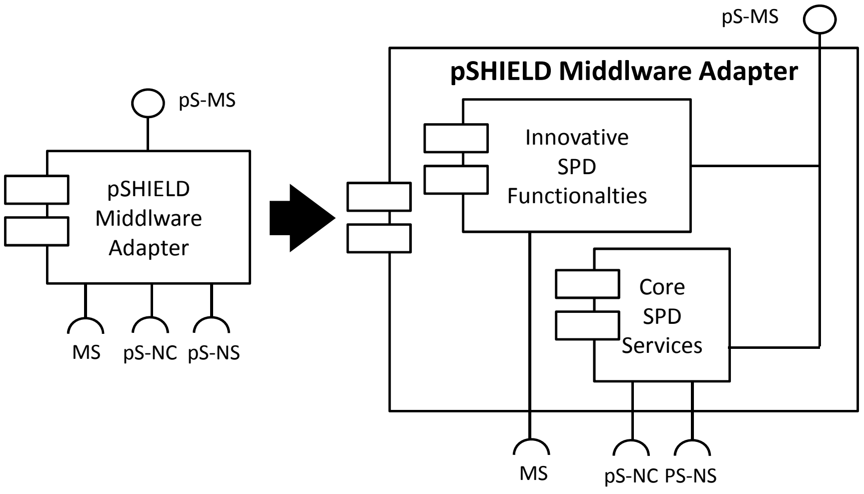

pS-MS
pSHIELD Middlware Adapter
MS
pS-NC
pS-NS
pS-MS
pSHIELD Middlware Adapter
Innovative SPD Functionalties
Core SPD Services
MS
pS-NC
PS-NS

Figure 47 – pSHIELD Middleware Adapter component architecture.

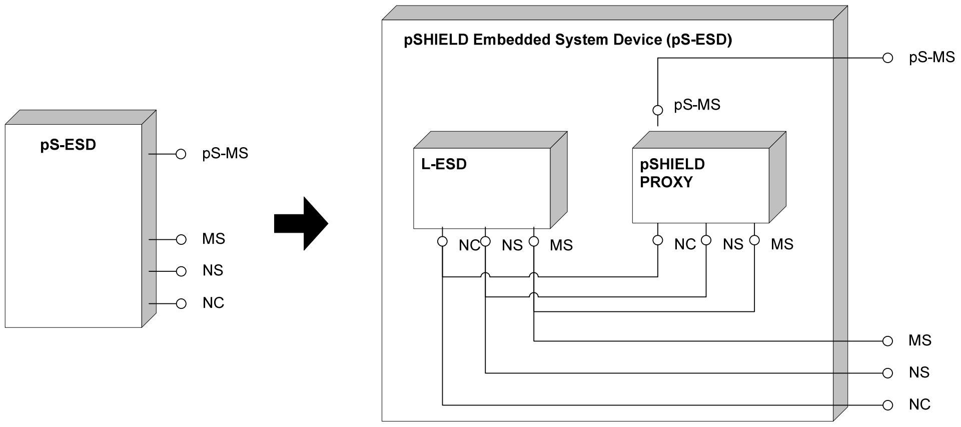


Figure 48 – pS-ESD component architecture.

Finally the Legacy Embedded System Device (L-ESD) can be further exploded into a more detailed component diagram, as shown in corresponding figure below.

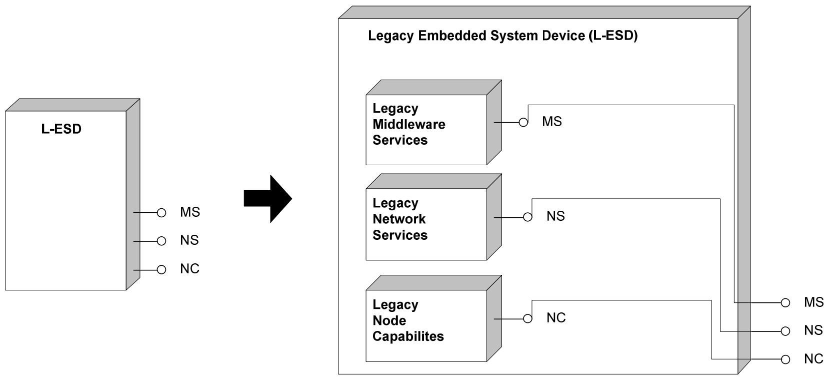


Figure 49 – L-ESD component architecture

The L-ESD architecture is composed by three components: (i) the Legacy Middleware Services; (ii) the Legacy Network Services and (iii) the Legacy Node Capabilities.

The **Legacy Middleware Services** includes all the legacy middleware services (i.e. messaging, remote procedure calls, objects/content requests, etc.) provided by the Legacy Embedded System Device which are not pSHIELD-compliant. In order to be pSHIELD compliant, these services should be enriched with pSHIELD SPD functionalities. The pSHIELD Middleware Adapter is in charge of this task.

The **Legacy Network Services** includes all the legacy network services (protocol stacks, routing, scheduling, Quality of Service, admission control, traffic shaping, etc.) provided by the Legacy Embedded System Device which are not pSHIELD-compliant. In order to be pSHIELD compliant, these services should be enriched with pSHIELD SPD functionalities. The pSHIELD Network Adapter is in charge of this task.

The **Legacy Node Capabilities** component includes all the legacy node capabilities (i.e. battery, CPU, memory, I/O ports, IRQ, etc.) provided by the Legacy Embedded System Device which are not pSHIELD compliant. In order to be pSHIELD-compliant, these capabilities should be enhanced and enriched with pSHIELD SPD functionalities. The pSHIELD Node Adapter is in charge of this task.

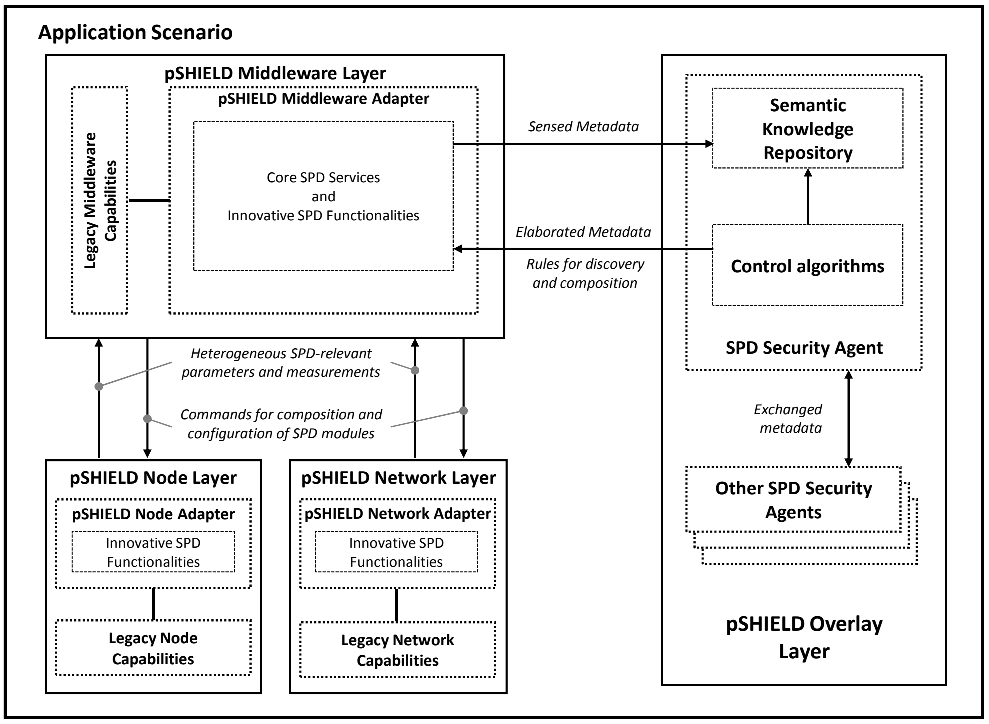

Application Scenario
pSHIELD Middleware Layer
pSHIELD Middleware Adapter
Legacy Middleware Capabilities
Core SPD Services and Innovative SPD Functionalities
Sensed Metadata
Elaborated Metadata
Rules for discovery and composition
Semantic Knowledge Repository
Control algorithms
SPD Security Agent
Heterogeneous SPD-relevant parameters and measurements
Commands for composition and configuration of SPD modules
Exchanged metadata
pSHIELD Node Layer
pSHIELD Node Adapter
Innovative SPD Functionalities
Legacy Node Capabilities
pSHIELD Network Layer
pSHIELD Network Adapter
Innovative SPD Functionalities
Legacy Network Capabilities
Other SPD Security Agents
pSHIELD Overlay Layer

Figure 50 – pSHIELD functional component architecture.

The result of the above considerations is the pSHIELD functional architecture depicted in Figure 50. This figure highlights some of the key pSHIELD enabling technologies mapped into the four functional layers introduced in the previous section. More specifically:

1. The pSHIELD Node

2. The Innovative SPD functionalities (e.g. smart SPD driven transmission and Cryptographic algorithms)
3. The semantic knowledge models
4. The core SPD services
5. The control algorithms based on Hybrid Automata theory

6.6 Interfaces

This section attempts to register and to categorize the interfaces which occur in a pSHIELD system. The term is quite generic, but the importance is great, since the interfaces are used to determine the interconnectivity internally, as well as the presentation and application capabilities of a platform. Consequently, the categories indicated hereafter may include overlapping elements, considering the fact that the definition is dependent on the different level of approach regarding “interface”.

6.6.1 Internal

The interfaces between the four pSHIELD layers are registered below:

Overlay interfaces

Interfaces used for the interconnection with Middleware layer:

1. Sensed Metadata (transferring information to Overlay)
2. Elaborated Metadata (transferring information to Middleware)
3. Rules for discovery and composition (transferring information to Middleware)

Middleware interfaces

Apart from the interfaces with Overlay (mentioned just above), two interfaces handle the SPD related data exchange with the underlying layers:

1. Heterogeneous SPD relevant parameters and measurements (from Node and Network to Middleware)
2. Commands for composition and configuration of SPD modules (from Middleware to Node and Network)

Network interfaces

The family of network interfaces is constituted from those registered in § 6.2.5 (NS, pS-NS), along with all the devices or parts of equipment serving as demarcation points in telecommunication networks lying in every pSHIELD implementation (Ethernet, Infiniband interface in Power Node etc.)

Node interfaces

A pSHIELD node exchanges data, measures parameters and receives commands through network, middleware and overlay interfaces, which ensure the communication with the respective incumbent layers.

6.6.2 External

The interfaces between the systems and the applications, along with those for (remote) administration of a pSHIELD system are two cases of external interfaces:

Remote commands

The interface through which a node receives configuration commands, data and equipment status form an external operator.

SNMP interface

An external maintenance system can perform node monitoring and control through an SNMP interface.

User interfaces

The interfaces used in specific applications for the intercommunication between components or between the user and the system. For example, Shepherd (provided by Telenor Objects) is an M2M communication platform, on which pSHIELD nodes can be connected through APIs and transmit their sensor information securely. More detailed information about application information is being developed and presented in the framework of WP6.

6.6.3 Node

The interfaces that the nodes of the systems are using and providing are mentioned below:

I/O interfaces

Interfaces used to connect to peripherals and pSHIELD functionalities. Depending on the node type and functionality they can be RS232, Serial, GPIO etc.

Dedicated functions/modules interfaces

Interfaces (usually in master-slave form) which transmit the output of a dedicated function:

1. Health monitoring
2. Security
3. Application processor
4. Power management
5. Memory
6. Reconfiguration controller

Communication interfaces

Interfaces used for the networking or radio function of the node (Ethernet, USB, Wi-Fi, ZigBee, TPM and Infiniband).

6.6.4 Components

The interfaces between the components of the architecture are indicated below:

L-ESD interfaces

Three interfaces are exported by legacy embedded devices, representing the inter-layer communication of different components:

1. Middleware services interface, MS
2. Network services interface, NS
3. Node capabilities interface, NC

pS-ESD interface

The transition from legacy devices to the so called pSHIELD embedded device comes with the addition of interface pS-MS, providing pSHIELD functionalities at Middleware layer. Moreover, pS-NC and pS-NS are the expansion of NC and NS (above) to include pSHIELD compliant functionalities.
These interfaces include the following features:

1. Function (operation of the component, further discriminated in name, input and output parameters)
2. Contract (used by the Core SPD services to synthesize the available services beam)
3. Description (semantics)
4. SPD status (current SPD level)
5. Connector (connecting two functionalities)

pS-SPD-ESD interface

pSHIELD SPD Embedded System Device is outfitted with a minimal set of Overlay functionalities. This extra feature corresponds to the extra overlay interface pS-OS. The Overlay services are provided by Security Agent, the pSHIELD module responsible for the composability of the platform.

Security Agent interfaces

Two internal interfaces (inside every Security Agent) are used to exchange information between the two modules of SA. They are pS-SKR and pS-CA, arranging the communication between Semantic Knowledge Representation and Control Algorithms modules.

Semantic models

The Semantic models are responsible for the efficient interpretation of syntax of data structures and knowledge in the application domains. Two interfaces are realized in these components, namely interface DB, between the Semantic Reasoner and the Database and interface KR (Knowledge Repository), between semantic aware components.

7 Conclusion

D2.3 represents consortium's effort to conclude on a preliminary, as complete as possible though, system architecture. Special care was given, to include all system design relating factors impact, as explained in the Executive Summary of this document. A formalization methodology was followed, to derive the architecture, from a cooperation of known pre-defined components and functions (that we call legacy) with modules that possess features and capabilities representative of pSHIELD project application domain. In this hierarchy of adding functionalities and value, three main subsystems were introduced: the Legacy Embedded System Device (L-ESD), the pSHIELD Embedded System Device (pS-ESD) and the pSHIELD SPD Embedded System Device (pS-SPD-ESD). The pSHIELD System Architecture, from the view of Embedded System Device (ESD) types, is practically consisted from the possible networking versions of the aforementioned subsystems. pSHIELD team has been working towards refining the architecture proposal and present an optimized system solution with the deliverance of D2.3.2.

8 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] pSHIELD, D2.1.2 “System Requirements and Specifications”
[32] Akyildiz and Jornet, Georgia Institute of Technology, “The Internet of Nano-Things”, IEEE Wireless Communications