Project no: 269317

nSHIELD

new embedded Systems arcHItecturE for multi-Layer Dependable solutions

Instrument type: Collaborative Project, JTI-CP-ARTEMIS

Priority name: Embedded Systems

D4.3: Preliminary SPD network technologies prototype report

Due date of deliverable: M18 –2013.02.28

Actual submission date: M18 - 2013.02.28

Start date of project: 01/09/2011 Duration: 36 months

Organisation name of lead contractor for this deliverable:

University of Geneva, UNIGE

Revision [Version 3.0]

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**		
Kresimir Dabcevic	UNIGE	11/01/13	
Lucio Marcenaro	UNIGE	11/01/13	
Virgilio Esposto	SES	18/01/13	
Ignasi Barri Vilardell	ISL	18/01/13	
Roberto Uribeetxeberria	MGEP	20/01/13	
Inaki Arenaza	MGEP	26/01/13	
Andreas Papalambrou	ATHENA	01/02/13	
Iñaki Eguia Elejabarrieta	TECNALIA	04/02/13	
Konstantinos Rantos	TUC	04/02/13	
Georgios Hatzivasilis	TUC	04/02/13	
Alexandros Papanikolaou	TUC	04/02/13	
Dimitris Geneiatakis	TUC	04/02/13	
Harry Manifavas	TUC	04/02/13	
Iñaki Eguia Elejabarrieta	TECNALIA	04/02/13	
Nikos Pappas	HAI	05/02/13	
Luca Geretti	UNIUD	20/02/13	
Antonio Abramo	UNIUD	20/02/13	
Kiriakos Georgouleas	HAI	20/02/13	
Eva Vázquez de Prada	ISL	20/02/13	
Ester Artieda Puyal	ISL	20/02/13	
Arkaitz Gamino	TECNALIA	22/02/13	
John Gialelis	ATHENA	22/02/13	
Reviewed by			
Name	**Company**		
Approved by			
Name	**Company**		

Applicable Documents		
ID	**Document**	**Description**
[01]	TA	nSHIELD Technical Annex

Modification History		
Issue	**Date**	**Description**
V1.0	11.01.2013	Draft version of the document, by UNIGE
V1.1	12.01.2013	Contribution to section 2, by UNIGE
V1.2	18.01.2013	Formatting and contribution to section 5, by ISL
V1.3	20.01.2013	Contribution to section 4.1, by MGEP
V1.4	01.02.2013	Contribution to section 3, by ATHENA
V1.5	04.02.2013	Contribution to section 5.2, by TECNALIA
V1.6	04.02.2013	Contribution to sections 4.3 and 5.1, by TUC and HAI
V1.7	05.02.2013	Contribution to section 4.2, by HAI
V1.8	20.02.2013	Modification of section 2, by UNIGE; formatting of the whole document
V1.9	20.02.2013	Contribution to section 3.2, by UNIUD
V2.0	20.02.2013	Consolidation of chapter 4, by HAI
V2.1	20.02.2013	Refinement of IDS algorithm, MGEP
V2.2	20.02.2013	Secure communication protocols, INDRA
V2.3	22.02.2013	Modification to section 3.1 by ATHENA
V2.4	22.02.2013	Contribution from TUC in sections 4 and 5
V2.5	23.02.2013	Contributions' merging
V2.6	24.02.2013	Refinement
V2.7	25.02.2013	Further Refinement
V2.8	25.02.2013	Modification of section 5.1 by INDRA
V2.9	26.02.2013	Move certification creation to D4.2
V2.10	26.02.2013	Refinement of IDS algorithm, MGEP
V2.11	26.02.2013	Expanded contribution to section 3.2, by UNIUD
V2.12	26.02.2013	Expanded ATHENA contribution
V3.0	28.02.2013	Final formatting

Executive Summary

This deliverable is focused on the detailed description of the network technologies that are currently under development in work package 4, conforming to the preliminary architecture and the composability requirements specified in deliverables D2.4 and D2.5. These technologies will be made available to the application scenarios and can be used as building blocks for the project demonstrators. This deliverable will be updated and refined in the second part of the project based on the final requests received from the application scenarios and on the refined system architecture, metrics and composition strategy to be followed.

Contents

Figures

Tables

Algorithms

Glossary

Please refer to the Glossary document, which is common for all the deliverables in nSHIELD.

1 Introduction

The nSHIELD project proposes a layered architecture to provide intrinsic SPD features and functionalities to embedded systems. In this layered architecture, building on top of the node functionalities defined in the WP3, Work Package 4 deals with implementation of the SPD functionalities at the network layer.

The workload encompassed in work package four is divided into four complementing work tasks:

- T4.1 Smart Transmission Layer;
- T4.2 Distributed self-x models;
- T4.3 Reputation-based resource management technologies;
- T4.4 Trusted and dependable Connectivity

Each of the tasks places focus on independent development and application of different SPD technologies at the network layer. As such, the technologies' performance will at first be evaluated on an individual basis, categorized with respect to their complexity and suitability for the proposed SPD levels and capabilities of different node classes. This will provide an output useful for merging the contributions into a system consisting of a set of mutually-collaborating approaches.

Each of the partners involved in the WP4 – coming from different backgrounds - brings to the table their own expertise and work style, often resulting in usage of different technologies for algorithm development. At this stage of the project, however, certain decisions on common approaches had to be taken to ensure that the work is presentable in a clear, technically correct and understandable manner, as well as to ease the future integration of all the contributions.

These common approaches are namely:

- Creation of the "SPD level – node class" matrix in order to demonstrate applicability of each of the algorithms under development to different nSHIELD node classes and different levels of SPD controlled by the Overlay
- Usage of the pseudo-codes for presenting the algorithm functionality in a clear, readable way
- Decision on types of commercially available embedded nodes that may be commonly used for porting and testing of the developed algorithms

Deliverable D4.3 presents all the algorithms - and ideas and approaches behind them - in WP4 that have reached demonstrable level. As such, the deliverable is structured as follows:

1. Introduction – provides a brief introduction to the problematic dealt with in WP4
2. Smart SPD driven transmission – presents an architecture for the provisioning of the secure and reliable transceiving in harsh channel conditions, built upon the software defined radio platform interconnected with the Power node. Performances of the algorithm for countering Smart Jamming Attacks in centralized networks are shown.
3. Distributed self-x models – provides means for recognizing and categorizing different denial-of-service attacks. In addition, a cellular-automata based consensus algorithm is presented, explaining the means for self-reconfigurability of the elements in the network.
4. Reputation-based resource management technologies – focuses on state-of-the-art reputation systems for distributed wireless networks, providing the output for the deployment of secure routing protocols.
5. Trusted and dependable Connectivity – presents a set of lightweight algorithms for the nSHIELD constrained (Nano) nodes, as well as somewhat computationally more demanding algorithms for micro and power nodes. Furthermore, explains the premises behind the access control in Smart Grid networks.

As was stated, these algorithms may be classified according to their suitability to different node classes and according to the estimated SPD level they provide. The proposal of such a categorization is given in a following table (the table of algorithms referred to in the table can be found on page 8 of this document).

Table 1-1: Categorization of developed algorithms in with respect to node class and SPD level

SPD level / node class	NANO NODE		MICRO NODE	POWER NODE	
	SPD-emulated	**SPD-enabled**		**Standard**	**SDR**
1 (lowest)				Atta	STL - AJ: -
2 (low)					STL - AJ: A1
3 (medium)					STL - AJ: A1 + A2
4 (high)					STL - AJ: A1 + A2 + A3

This proposal is by no means definite, rather should serve as an overview of a direction we have taken towards deployment of different SPD functionalities at the network level.

As an example, Smart Transmission Layer (STL) is a functionality developed for the SDR-capable Power nodes only. Anti-jamming algorithm complexity depends on the SPD level imposed by the Overlay – in this case, for achieving anti-jamming (AJ) capabilities, algorithm A1 (Frequency switching) is deployed in case of SPD=2, whereas algorithms A1, A2 and A3 are combined in cases of SPD=4.

2 Smart SPD driven transmission

Smart SPD driven transmission is a service deployed at the network level, whose goal is ensuring smart and secure data transmission in critical channel conditions. For achieving this, concepts of Software Defined Radios and Cognitive Radios are being utilized.

The main enabler for achieving the Smart SPD driven transmission is the proposed and currently-developed Security-Aware Framework for Cognitive Radio Networks. The completed framework shall encompass the most suitable security solutions for each of the identified potential threats to Software Defined Radio and Cognitive Radio systems.

Crucial identified threats are the Smart Jamming Attacks (SJAs), Primary User Emulation Attacks (PUEAs), Spectrum Sensing Data Falsification Attacks (SSDFAs) and Objective Function Attacks (OFAs). Each of these threats first needs to be addressed separately, whereas ultimately a Security-Aware Framework based on Intrusion Detection System shall unify all of the proposed algorithms, ensuring the provision of SPD in SDR and CRN networks.

For demonstrating the functionality and effectiveness of the algorithms developed within the Security-aware framework, a proprietary C++-based simulator is being used. Up to date, an algorithm able to efficiently counter various Smart Jamming Attacks has been developed, details of which are introduced in section 2.2.

As useful as the simulating environment is for the algorithm research and development, simulators of wireless systems necessarily introduce many abstractions, often leading to losing track of important real-life constraints and obstacles. As such, demonstrating effectiveness of wireless system's cognitive features on a simulation basis only is not sufficient. For that purpose, a real-life test bed prototype was devised and assembled, allowing for evaluation of actual performance and behaviour, unattainable with simulation models alone.

Processing and communications loads, self-organizing and reorganizing in case of node crash, interference recognitions and countermeasures, on-the-air authentication, delays induced by multi-hop, network recovery in case of crash of nodes and intrusion detection are some of the parameters that must be evaluated.

Section 2.1 is dedicated to describing the prototype to be set-up for the verification of SPD driven Transmission.

2.1 Smart Transmission Layer test-bed prototype

2.1.1 Description

Talking about a cognitive wireless network, the following key issues are to be considered and properly implemented:

- Spectrum sensing
- Spectrum sharing
- Spectrum management
- Interference management
- Network establishment and maintenance
- Access techniques
- Routing protocols
- Security
- Energy efficiency

The main scope of the prototype is placed on exploring the issues related to spectrum sensing, spectrum and interference management, and security.

Other aspects like network establishment, access, routing, and energy efficiency are mainly delegated to the existing SDR radio platform and to the waveforms used in the prototype.

Wherever necessary for proper understanding of cognitive features, corresponding details of the underlying functionalities will be given.

2.1.2 Prototype setup

Prototype consists of three SDR wireless, each composed of a wireless hand-held (HH) terminal (Secure Wideband Multi-role - Single-Channel Handheld Radio) and an external processing platform representing the nSHIELD Power Node (OMBRA V2).

HH is a V/UHF SCA compliant SDR radio terminal capable of hosting multiple wideband and narrowband waveforms, spanning from simple PTT voice to meshed MANET, with different grades of embedded EPM and security features, with built-in GPS receiver.

OMBRA v2 is a powerful embedded platform featuring an OMAP processor @1Ghz, Texas Instruments' DSP processor @600 MHz and a Xilinx Spartan FPGA. The platform is capable of running either WinCE or Linux OS, has an on-board FLASH and RAM, and several peripherals (USB, Ethernet, serial, etc.).

A processor of the same family is also used in HH, making future porting/integration straightforward.

HH and OMBRA v2 are connected through a high speed serial connection for quasi-real-time spectrum acquisition, and through an USB/Ethernet for monitoring and control of the radio.

A carrier board provides electrical interfaces towards radio and external instrumentation, as well as the power supply regulation for OMBRA v2 board. Figure 2-1 sketches the architecture of the wireless node.

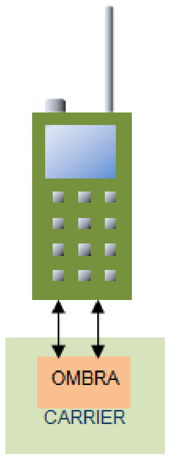


Figure 2-1: Wireless node architecture

2.1.3 Spectrum acquisition

HH receiver is fully digital. In VHF, analog to digital conversion is performed directly at RF while in UHF A/D conversion is performed at IF.

No selective filtering is applied before ADC.

Broadband digitized signal is issued to an FPGA where it undergoes digital down conversion, matched filtering and demodulation. The signal at FPGA input is a sample of the raw spectrum.

The raw samples are stored in a RAM buffer internal to the FPGA and output through a fast serial port to the OMBRA board, where they are processed.

Due to the high speed of the ADC (250 MHz), serial port speed is not sufficient for true real-time transfer; in addition processing capabilities of OMBRA board would be totally devoted to the processing of received signal, leaving no room for higher level applications. Furthermore, power consumption would be heavily as well.

Adopted solution consists of performing a quasi-real-time acquisition, i.e. collecting a large “snapshot” of incoming spectrum; say tens of kilo-samples, and to transfer the snapshot to OMBRA board.

Once the snapshot has been transferred, spectrum acquisition process re-starts.

This is sufficient for proper analysis of the majority of RF scenarios: only fast pulsed signals might be completely missed.

Future hardware enhancements will likely make the full real-time acquisition possible.

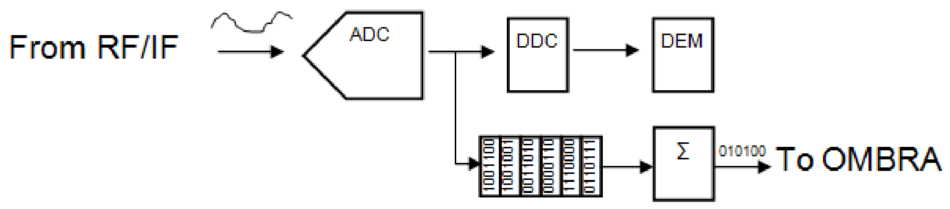


Figure 2-2: Spectrum acquisition process

2.1.4 System architecture

The prototype system will be composed of three wireless nodes.

The nodes may be connected in full-mesh fashion, PmP, multi-hop PtP, single hop PtP with the third node acting as an interferer/intruder. Those arrangements cover all of the basic network topologies.

The following figure provides a sketch of the possible network arrangements.

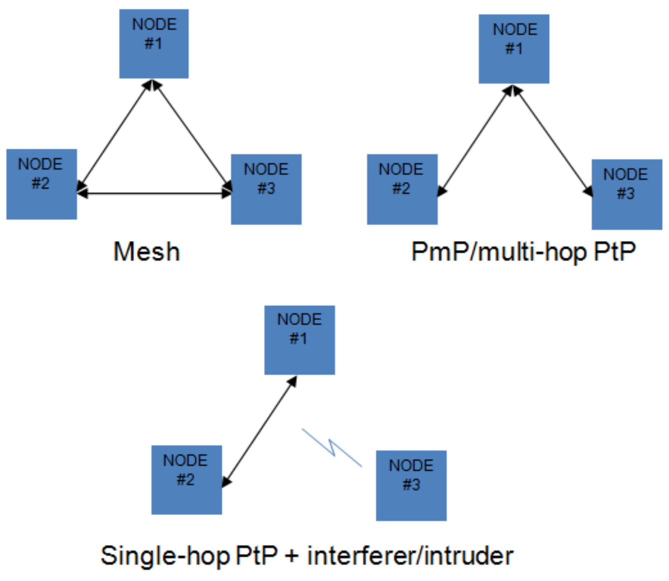


Figure 2-3: Basic network topologies attainable with the prototype

In order to simulate different scenarios, there has to be a possibility of dynamically changing nodes’ connectivity, as well as injecting noise or interfering signals in a controlled and repeatable way.

Neglecting multipath effects, radio link can be simulated by a combination of passive RF components such as attenuators, directional couplers, splitters and combiners.

Simulated RF bench exhibits several advantages:

- accurate and stable RF levels can be set
- test instruments and generators can be connected to one or more branches

- complex dynamic behaviours of the transmission channel can be mimicked
- tests can be replicated without the typical uncertainties of over-the-air transmission

In the frequency range of interest, it is convenient to implement a coaxial RF bench. The following figure shows the schematic diagram of the proposed RF bench. Directional coupler(s) (in grey) can be connected in different positions in order to extract/inject signals from/to the various nodes.

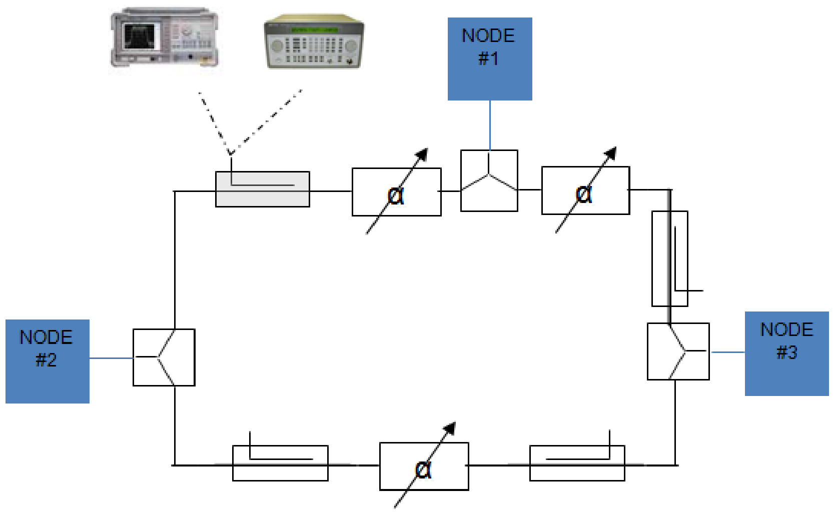


Figure 2-4: Scheme of the Smart Transmission Layer test-bed

2.1.5 Cognitive capabilities

The described test-bed will allow for testing the cognitive functionalities of the system. Talking about cognitive functionalities, the following will be developed:

- Self-awareness: the network learns the current topology, the number and identity of the participants, their position, and reacts to variations of their interconnection
- Spectrum awareness: the node collects information of the spectrum occupancy, either by spectrum sensing or indirectly using geolocation/database method, and shares the data with other nodes in order to create a map of existing sources
- Spectrum intelligence: the node analyses the spectrum information focusing on specific sources, trying to identify them by extracting significant signal parameters
- Jamming detection and counteraction: the nodes recognize the presence of hostile signals and inform the rest of the network’s nodes; the nodes cooperate in order to come with the optimal strategy for avoiding disruption of the network
- Self-protection: the nodes cooperate to detect and prevent the association of rogue devices and to reject nodes that exhibit illegal or suspicious behaviour.

While testing the self-awareness, the following networking functionalities will be exercised:

- Network entry
- Node authentication
- Internode communication
- Topology awareness: number of nodes, mutual visibility, connection, location
- Reconnection of a node after power cycle/link loss
- Choice of operating frequencies in accordance with predefined frequency plan

Verification of the spectrum awareness capability will consist in system taking “snapshots” of the electromagnetic environment and comparing them to a database of the known sources, relating the information with the geolocation of the single nodes and network topology to detect unexpected/unknown signal sources.

A map of the existing sources will be generated and made available to system management for further analysis.

In the simplest case the map is a simple list of detected sources; correlating received power and time of arrival at the various nodes, an attempt can be made to localize the sources on a physical map.

Spectrum intelligence refers to the node performing the analysis of specific waveforms in order to extract significant information (such as cyclostationary features) useful for identifying them. Ideally, the signal should be fully identifiable by its parameters: carrier frequency, bandwidth, transmit technique (FDD, TDD), modulation scheme, bit rate, and so on. This kind of analysis is very demanding in terms of algorithmic and computational power. Heuristic methods, based on database of known signature would be useful to speed up and ease the processing and provide acceptable results. The system should be able to interact with upper layers to inform of the existence of unidentified signature and possibly for processing supplement with the aid of external resource.

From measurement of link parameters (received power, signal quality, BER / PER) the wireless node can infer the presence of a jammer and react accordingly, moving to a new frequency, changing physical or logical waveform parameters (modulation, bit-rate, TX power, FEC, MAC protocols), or selecting a different waveform at all, depending upon the capability of the SDR and the nature of the offending signal.

Additional strategies can be adopted at network level, e.g. modifying routing around the affected node(s), in order to mitigate the overall effect of jamming. More in-depth explanation of anti-jamming algorithms is given in the section 2.2.

Self-protection is a feature related to the capability of providing secure and reliable communication to the subscribers. Several protection mechanisms are available at different layers of the node:

- Encryption and authentication operated directly by the SDR provide basic security layer to the node.
- Encryption keys and passwords may be modified according to several criteria defined at network level or at supervisory level in order to protect against the intrusion of rogue terminals.
- Higher degrees of authentication / encryption can be adopted at upper layers to offer further protection to users/subscribers as well as to the network infrastructure.

Behaviour of the nodes may be monitored, checking for abnormal operation like repeated attempts to access to protected area, generation of excessive traffic/ band request, anomalous reporting of status information (e.g. node provides geolocation data not coherent with network topology), or rejection of requests coming from neighbours, etc. Each node may have a reputation score associated to it; the score is updated following criteria defined at network/supervisory level. A node can be expelled from the network if its score exceeds certain thresholds. Considered reputation mechanisms are discussed in section 4.

2.1.6 Demonstration Scenarios

In terms of applicability to the nSHIELD common demonstrators, Smart Transmission Layer finds its place within the Dependable Avionics Scenario (T7.3). More details on implementation of Smart Transmission Layer within the said scenario will be presented in deliverable D7.11.

2.2 Algorithm for countering Smart Jamming Attacks in centralized networks

2.2.1 Description

Improved radio capabilities of Cognitive Radios also bring advanced possibilities with respect to attackers’ actions and complexity levels, starting with the possibility of jamming multiple frequencies and larger frequency bands by self-reconfiguring their transmission parameters ”on-the-fly”. Such advanced jammers, operating in SDR Networks and CR Networks are from now on referred to as “Smart” or “Intelligent” jammers [1].

The proposed scheme for countering SJAs consists of three complementary mechanisms: Frequency Switching, Reputation [2], and Trajectory Altering. Each of these mechanisms will be explained in greater detail as follows:

Frequency switching occurs whenever a radio node finds itself under the jamming influence. The process is initiated by the Central Entity, which decides on frequency switching based on the observed link quality. CE, which has a continuous access to the pool of available frequencies, randomly selects a non-jammed frequency and instructs the radio node to initiate the frequency switching process. This is done over a dedicated control channel. For now, it is assumed that this channel is unsusceptible to jamming disturbances, therefore the frequency switching can always take place (in the future, threats to dedicated channel shall also be considered).

```
Algorithm 1 Frequency Switching mechanism
for (all the registered nodes in the networks) do
   if (observed node's link quality is lower than a pre-defined threshold)
then
      (obtain its current frequency Ftx)
      (choose a frequency from a pool different than Ftx)
      (assign the new frequency to an observed node)
   end if
end for
```

Algorithm 1: Frequency switching mechanism

Reputation algorithm is used for keeping track of the radio nodes that repetitively create jamming disturbances. Upon entering the network, every node is given an equal initial reputation value. As jamming occurs in a particular area under CE’s supervision, it is important to figure out the cause of the jamming. However, since it is impossible for the CE to exactly single out the jamming source in a given area, it decreases the reputation of all the nodes that are currently at that approximate location. After a node has collected a certain number of negative reputation points, it is being deemed by the Central Entity as a jammer, its future observations are not being taken into account anymore, and the CE starts alerting all the radio nodes that are approaching it of its presence.

```
Algorithm 2 Reputation mechanism
for (the node that is currently observed as being jammed) do
   for (all the nodes within observed node's sensing radius) do
      (deduce a frequency point)
   end for
end for

for (all the registered nodes in the networks) do
   If (overall reputation is under a pre-defined threshold) then
      (mark as a jammer)
      (inform all the other radio nodes of jammer's geographical position)
   end if
end for
```

Algorithm 2: Reputation mechanism

Once the jamming source has been identified, several potential actions are available to an intelligent Centralized Entity. Whereas one of the potential actions is excluding the jammer from the network, thus preventing it from using network's resources, this action doesn't prevent the jammer from pursuing its malicious behaviour. Hence, a better choice is keeping this entity in the network, continuing to receive its data, and using its geolocational observations to suggest an altered trajectory to the nodes that are approaching it. As the jamming node changes and updates its location, so does the suggested trajectory for the approaching nodes.

```
Algorithm 3 Trajectory-altering mechanism
for (the node marked as a jammer) do
     (obtain its geographical position)
     for (all the nodes in the network except for the jammer) do
         (obtain their geographical positions)
     If (any of them are within a certain distance from jamrner's estimated
         jamming area) then
        (suggest a next safe point to move to)
     end if
   end for
end for
(obtain updated geographical positions of all the nodes, including jammer)
```

Algorithm 3: Trajectory-altering mechanism

Naturally, overall capabilities of potential jammers can vary. Following that, five different potential jammers with increasingly advanced capabilities have been modelled to demonstrate effectiveness of the scheme, namely:

- Naive jammer - jams only one frequency, continuously, and has no information regarding other radio nodes;
- Random jammer - is able to jam multiple frequencies, but chooses which frequencies to jam randomly, every predefined number of timeframes, and has no information regarding other radio nodes;
- Adaptive jammer - senses which frequencies are being used by the surrounding nodes, and reconfigures its jamming parameters in order to start jamming the sensed frequencies;
- Reputation attacking jammer - besides inheriting the capabilities of Adaptive jammer, also has the ability to emulate being jammed by the surrounding nodes with a certain probability, thus tricking the reputation algorithm into erroneously assigning negative reputation points to "good" nodes;
- Tracking jammer - Adaptive jammer which is able to alter its trajectories in order to follow and continuously jam a specific radio node.

For developing the algorithm and demonstrating the effectiveness of the proposed scheme, a C++-based demonstrator was built. The demonstrator allows for adding an optional number of each of the three node classes to the scene: "good" radio nodes (Default: 4), "jamming" radio nodes (Default: 1) and Central Entities (Default: 1). Each of the added jammers can have characteristics of one of the five jamming subclasses described above.

As previously explained, the proposed reputation mechanism assigns negative reputation points not only to the real jammers, but also to "good" nodes who find themselves close to a jammed node while jamming occurs. Since this might potentially lead to a wrong decision with respect to which nodes should be marked as a "jammers", it is important to demonstrate effectiveness of the scheme through the simulated probability of correct detection.

Table 2-1: Probability of correct jammer detection

Jammer type	Prob. of correct detection
Naive	1
Random	1
Adaptive	1
Reputation attacking. Ps=25%	1
Reputation attacking. Ps=33%	0.92
Reputation attacking. Ps=50%	0.55
Reputation attacking. Ps=75%	0.15
Reputation attacking. Ps=100%	0
Tracking	1

2.2.2 Demonstration Scenarios

With respect to the considered nSHIELD scenarios, the developed algorithm has particular importance and applicability for scenario S3 (Dependable Avionic Systems), where ensuring continuous, fault-tolerant communication is paramount.

3 Distributed self-x models

3.1 Recognizing & modelling of denial-of-service attacks

3.1.1 Introduction

Denial of Service (DoS) attacks are attacks that attempt to diminish a network's capacity and resources and thus render it unable to perform its function, rather than attacking the actual systems they target. Resources that can be targeted typically include network capacity, processing resources, energy and memory. Usually a combination of these resources is affected as they are related, for instance increased network traffic will result in increased processing requirements and therefore energy.

Categories

A large number of DoS attack categories exists. The most important ones, related to the area of secure embedded systems, are Jamming, Tampering, Collision, Exhaustion, Unfairness, Neglect and greed, Homing, Misdirection, Egress filtering, Flooding and Desynchronization. Not all of these attacks can be related to the SHIELD framework. Some of these concern physical implementations, known as side channel attacks, or affect very specific protocol mechanisms that are beyond the architectural abstraction layer related to SHIELD. DoS attacks can often appear in the form of distributed attacks (known as DDoS) where the attacker is now a single node but rather a smaller or larger number of nodes aiming at disarming defence mechanisms capable to block attacks originating from single points.

Detection

In general, DoS attacks can be detected with methods centred on the concept of faults and abnormal behaviour.

Faults are events in the system that is clearly related to a malfunction of the system or a particular subsystem. Detected faults can be normal, part of routine malfunctions, or abnormal. These detected faults need to be distinguished against normal faults, especially when a large number of nodes exist.

Normal behaviour is defined as the set of expected operation status of the system and the subsystems. Behaviour needs to be monitored in order to be judged as normal or abnormal. Abnormal behaviour can include increased or decreased usage of resources (network, hardware etc.), unexpected patterns of events and more. Abnormal behaviour is distinct to a fault in that the systems continue to be operational, regardless of its possible inability to fulfil the requested services due to exceeding of capacity.

DoS attack detection is a field partially overlapping with intrusion detection in methods and techniques. It is however different in that a DoS attack can take place without any sort of intrusion or compromise.

Countermeasures

Denial of service attacks are in general complicated attacks regarding appropriate defence methods because there is usually no control over the source of the attack. Assuming successful detection of an attack, various countermeasures of different aggressiveness can be taken ranging from passive response such as blocking network traffic and waiting in order to save system resources, to active responses such as throttling traffic or tracing the source of the attack in order to maintain operation of the network.

Network architecture issues

DoS attacks can take place toward all layers including the physical, MAC, network, transfer and application. As a result, successful mechanisms against DoS attacks need to be implemented to various layers where protection is needed. Moreover, cross-layer mechanisms and inter-layer communication are necessary in order to detect and defend. When systems under attack are distributed rather than centralized, a distributed mechanism is needed in order to detect attacks.

3.1.2 Architecture

The proposed DoS attack detection scheme will involve both distributed intelligence and a certain degree of node-specific input. The reason for this is that during DoS attacks the sheer amount of incoming data can lead to a self-invoked DoS attack if attempted to be analysed at the whole. Therefore, the following architectural procedure is proposed in order to hierarchize the detection method, as depicted in the following figure:

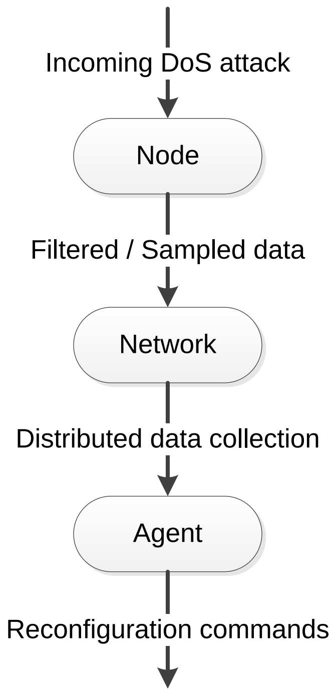


Figure 3-1: Architectural procedure for identifying DoS attacks

At the first level, incoming traffic is handled by a corresponding node procedure. This procedure takes place always, regardless of whether an attack is suspected or identified. The node performs a first-level sampling and filtering. Filtering refers to the recognizing of traffic that is known to be legitimate without forwarding it to the next level and sampling refers to the process of selectively forwarding packets to the next level at a predetermined sampling rate. This can provide a valid and representing input to the detection algorithm without overloading or transferring the load associated with the DoS attack.

The distributed algorithm running at the network level of the SHIELD architecture will perform the main detection task of the DoS protection. This will consist of statistical analysis and pattern matching analysis performed on the inputs provided by the node. Distributed data will be collected by the corresponding agent at the overlay level which will issue the reconfiguration commands based on the combined results provided by the nodes.

The success of the identification methods depend largely on the extent of deployment. A global deployment among all nodes would increase the success of the detection. However, often it is not possible to deploy globally because of various reasons such as the coexistence of nodes belonging to different classes or network topology limitations. Therefore, deployment is usually not global but local as depicted in the following figure.

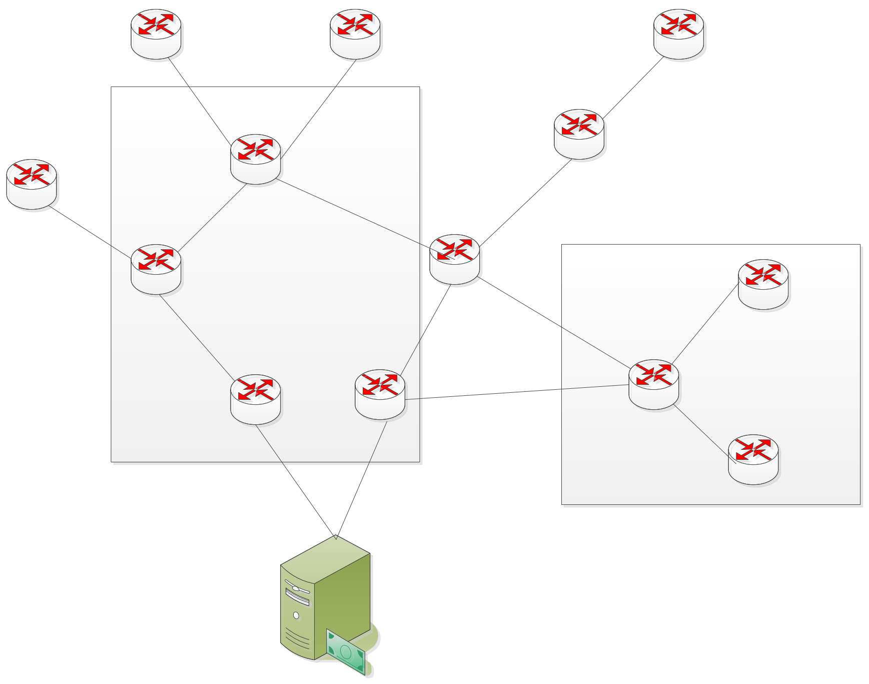


Figure 3-2: Typically, deployment of the algorithm is not global but local in various administrative domains depicted in the rectangle areas.

3.1.3 Overview of inputs

As previously mentioned, the inputs will be initially filtered and sampled at the node level at a varying degree depending on the system status and the corresponding input. The main inputs to the DoS detection scheme are proposed to be the following:

Power consumption

Power consumption is a critical aspect for each node's operation, especially for wireless and minimal resource nodes. The power modules of nodes can be one of the most important targets for DoS attacks. This can take place by sending too many requests, sending resource-intensive requests or causing faults that lead to increased power consumption. The power consumption data is provided by the power module of the node. An appropriate sampling rate is needed that will depend on the sampling rate of other inputs, with which it will need to be correlated, as well as the type of node. In general, power consumption data are of minimal size and can be sampled at increased rates. Power consumption is important because it can be correlated to normal or abnormal operation. Power consumption can be measured and be known beforehand as well as its expected deviations for normal operation. Patterns and values outside of these ranges can trigger a DoS attack flag.

CPU usage

CPU usage has been correlated to DoS attacks. One of the ways DoS attacks work is to cause excess computational load to the node that may or may not be correlated to a respective increase in traffic. This in turn can lead to a depletion of power resources however it is a distinct input than power consumption because the latter of also largely affected by physical transmission.

Network traffic

Network traffic is an important input to the detection process since this information can help identify traffic patterns. Of course, not all network traffic can be forwarded but rather sampled data as well as first-level statistics. The sampling rate of packets can be variable depending on the actual amount of traffic. Moreover, easily distinguishable traffic patterns such as traffic spikes can be reported by nodes.

Number and identity of nodes interacting

Besides the actual traffic, it is important to take into account the amount of nodes interacting as well as their identity. The number of nodes interacting should correlate with the traffic pattern else this could be a sign of an attack taking place.

Signal strengths and transmission parameters

Various transmission parameters such as signal strength and transmission mode can also be used in the correlation processing.

The DoS attack algorithm can use the jamming detection and intrusion detection results as inputs to be correlated and included in the algorithmic process.

3.1.4 Overview of outputs

The outputs of the detection process can provoke actions that range from passive to more active methods.

DoS attack Flag

The most basic output of the detection process is a flag (or alert) that a DoS attack is detected or suspected.

Type identification

This refers to the identification or the type of attack, the level at which is takes place and the vulnerability it exploits. It is assumed that known DoS attack types are modelled in advance.

Traffic differentiation information

This refers to information describing the network traffic that forms part of the attack. This can help identify and differentiate legitimate traffic from the attacker's traffic and therefore be used for traffic mitigation. Mitigation refers to the throttling of inbound traffic based on identified types of attacker traffic, regardless of the knowledge of the actual source of the originating traffic.

Tracing information

Tracing is the most difficult task and output to be produced. Most often, tracing the source of a sophisticated DoS attack is not possible however this can be one of the outputs for less sophisticated attacks or attacks that have previously taken place and data exists for them.

3.1.5 Overview of detection method for Denial-of-service attacks

The proposed detection scheme combines and correlates the information provided at the inputs in order to provide the intended outputs and their associated actions. The methods used to detect and analyse the attacks are composed of two different algorithmic methods.

Statistical analysis algorithm

This algorithm intends to correlate all inputs in order to detect that an anomaly is taking place. Existing advanced statistical algorithms exist that can detect anomalies using comparison of samples to known probability distributions such as the Kolmogorov–Smirnov test. However, this becomes very difficult when it related to multiple parameters, multiple inputs and varying sampling rate. As a result, the proposed scheme is initially intended to use simpler but multi-parameter statistical correlations.

Pattern matching algorithm

This algorithm tries to detect patterns in the provided traffic that matches pre-determined attack patterns. These patterns, usually known as signatures, have to be previously generated based on simulations or from past attacks. As a result, the process of detecting DoS attacks in this way is strongly dependent of DoS attack modelling which is a discrete process that takes place previously.

An overview of the simplified operation of the algorithms is shown in pseudo code below:

Algorithm 4 DoS Attack Detection

```
#begin signature matching
#read inputs
read (communication packets);
sample (traffic);
read (signature_database);

#perform database matching
if (sampled traffic packet in signature database)then
   patern_matching_detection = true;

#begin statistical analysis
#read inputs
Repeat
{
  read (communication packets);
  read (power unit status);
  read (CPU status);
}

#perform analysis
analyze (power_consumption);
return power_consumption_score;
analyze (CPU_usage);
return CPU_usage_score;
analyze (network_traffic);
return network_traffic_score;
add (power_consumption_score + CPU_usage_score +
network_traffic_score) = total_score;

if (total_score > detection_threshold) then
   statistical_analysis_detection = true;

#combined matching
if {anyof(patern_matching_detection, statistical_analysis_detection) =
true} then
   raise DOS attack flag;
```

Algorithm 4: DoS attack detection

3.1.6 SPD level, node classes and demonstration scenarios

DoS attack detection can be seen as a parallel operation of two algorithms, the statistical analysis algorithm (SAG) and the pattern detection algorithm (PAM). These can provide independently their detections and a detection strategy can be chosen depending on the available resources. The pattern detection algorithm is based on pre-computed patterns therefore its operation of comparing patterns is less resource intensive and can therefore be used in a sufficiently capable micro node. The statistical analysis algorithm, on the contrary, has to perform real time complex statistical operations and therefore can be used in the power nodes, either standard or SDR enabled. Regarding the SPD level imposed in the power node, parallel operation of both algorithms can be enabled at the high (4th) and medium (3rd) SPD level and only one algorithm at the lower levels.

To develop and demonstrate the DoS attack detection scheme, simulators in the OMNET++ platform are being built. The effectiveness of the algorithms under development can be demonstrated by invoking various types of pre-selected DoS attacks and examining the detection rate. This can be related to most of the nSHIELD demonstration scenarios.

3.2 Model-based framework for dependable distributed computation

Dependability in terms of computation means the enforcement of two major properties on the application and the system that runs it:

- The application is defined in a way that some guarantees over its execution can be introduced;
- The application is able to tolerate faults in the system that runs it.

In particular, the guarantees of an application usually fall into the following categories:

1. Data integrity: received data is still correct after being transmitted over a medium;
2. Code correctness: data conversions and code semantics operate as expected or at least the designer is able to observe the behaviour of the application both at the debug and production phases;
3. Code authentication/authorization: code can be identified as coming from a trusted source, with an unambiguous versioning scheme that prevents an incorrect deployment.

Our model-based application framework (called *Atta*, after the eponymous ant genus) is designed to provide such guarantees by means of a distributed middleware that handles deployment, scheduling, computation and communication between a set of nodes. To achieve such result, the application is written according to a dataflow metamodel that enhances dependability in both the development and production phases.

Given an existing application, we can partially rewrite it in order to express it as a graph where vertices are code portions and edges are data dependencies. Then, if the hardware platform offers some redundancy, Atta is able to guarantee that node faults do not prevent the application from running to completion (or to provide a service in a continuous way). This feature is obtained by changing the execution place of code portions or by discarding unavailable data. If no redundancy is available, the framework still helps during the deployment phase, since code is automatically distributed from repositories to the nodes that participate in a computation task. In particular, the signature of the repository can be verified for security purposes. Also, if the application is not monolithic, its metamodel allows the definition of specific graph edges where data can be encrypted: this approach reduces the use of secure/private communication to the actual minimum required by the application.

On the other hand, when an application must be designed from the ground up, Atta provides several advantages. The first one is that both a top-down and a bottom-up approach can be performed: the metamodel easily allows to test a sub graph of the application and to refine it as soon as a more detailed implementation is needed. Other advantages stem from the methodology required to design applications in Atta. Since all artifacts (implementations and data types) must be versioned and outputs are organized into pnodes, a designer can always trace back his simulation results to the specific application that generated them. This is in contrast with the too-common development scenario where simulation/testing are performed on un-versioned (or monolithically versioned) software that produces unorganized, often overlapping, data. It can be seen that this holistic approach to development improves the quality of the software and reduces issues in the production phase, thus adding to the overall dependability of execution.

3.2.1 Framework architecture

Atta is designed to enhance both the development and production phases of an application. The development phase is concerned with dependable design, while the production phase with dependable execution. Here we mainly focus our attention on the production phase, but many concepts (like the application metamodel) also directly impact the development phase.

We envision a generic distributed application as being run on a set of *worker nodes* (here called "wnodes"), that are embedded nodes capable of accepting workload. Not all worker nodes necessarily perform computation at a given moment, some of them being redundant resources that may be called upon when a computing wnode fails.

In addition to wnodes, other sets of nodes that are relevant within Atta are the following:

- Synchronization nodes ("snodes"): nodes that have the task of synchronizing communication between worker nodes; their primary role is that of triggering an event to all of its listeners as soon as new data is available;
- Persistence nodes ("pnodes"): nodes that persist data; they are nothing more than a (distributed) database that wnodes can use to load/store data;
- Client nodes ("cnodes"): nodes that can inject or extract data for the application by leveraging the pnodes; they represent the external interface to the application, so they also allow composition of distinct applications;
- Directory nodes ("dnodes"): nodes that handle the participation of the other nodes to a specific application, also performing authentication and possibly authorization; they represent the authority within the framework, by grouping nodes and handling security;
- Repository nodes ("rnodes"): nodes that provide (parts of) the application to be deployed on the wnodes; they hold versioned repositories, i.e., Subversion or Git or Mercurial;

It must be noted here that an embedded node can participate on multiple sets and consequently can take on multiple *roles* within this software architecture. Clearly, multiple independent devices for each role are envisioned to improve the robustness of both the execution of the code and the consumption of the computed data. Consequently, the roles as defined above must be considered logical components of the runtime, rather than disjoint physical embedded platforms.

Multiple wnodes and snodes are particularly important to the runtime, in order to run to completion even in the presence of faults. Secondarily, multiple pnodes are preferable especially when data must be collected for later analysis. Multiple dnodes are not very important, since their concerns are mainly restricted to the setup phase of the execution of an application. However, their impact increases if the set of wnodes is very dynamic, as it is the case with mobile wireless networks. Multiple repository nodes make sense mostly due to the potentially significant data transfer required: if we offer mirrors, we can better balance the communication load, and also account for server downtimes. Finally, multiple cnodes instead are present only when multiple users interact with an application, hence they are not a replicated resource.

In Figure 3-3 the architecture of the runtime is shown, where a letter stands for a role (e.g., C for cnode), and an arrow starts from the initiator of a transaction and ends to the target of such transaction. We identify two important groups of roles that are involved in the *processing* and *setup* parts of the runtime:

- The processing part includes wnodes, snodes, pnodes and cnodes and it is rather straightforward: each wnode independently elaborates a code section (i.e., a vertex of the application graph), while getting/posting data (i.e., edges of the application graph) to the set of pnodes. The snodes are used to synchronize the wnodes by means of events. The cnodes may be used to inject or extract data at runtime by a user.
- The setup part is more complex: first, we need to instruct this distributed platform to actually perform the processing of a given application. This operation is clearly controlled by the user (from a cnode), who "instantiates" an application by submitting a job to the directory nodes. Then, depending on the policy used, wnodes enrol to a job or are asked to enrol by the dnodes. Such enrolment also informs wnodes/cnodes on the snodes/pnodes to use; in fact, snodes and pnodes, being server-like resources, are semi-statically chosen by the dnodes as services to the Atta runtime. The rnodes instead are completely static: the corresponding resources are encoded in the application metamodel itself. Wnodes are initially given a pointer to a repository that holds the highest-level information on the application. Then, a wnode can pull the information it actually needs for execution, thus performing deployment.

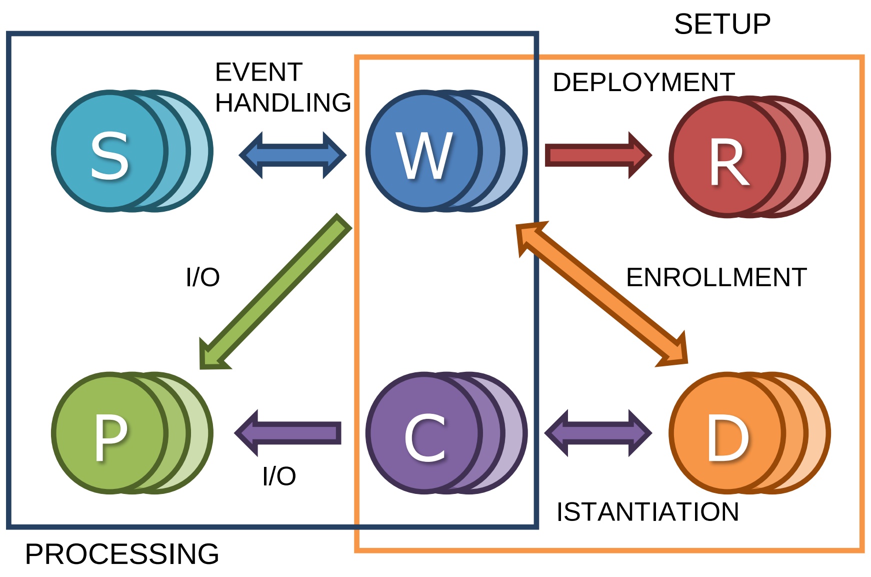


Figure 3-3: Architecture of the Atta runtime

3.2.2 Application metamodel

The architecture discussed above clearly needs a dedicated metamodel for the application.

While a monolithic application is certainly reasonable, on the other hand a composable application can actually exploit the distributed nature of the embedded network in terms of multiple resources that can operate concurrently. Consequently, the developed application metamodel relies on a model-based approach: the application is made of “blocks” that can contain either pure code or other blocks connected with each other. Due to this splitting of the application code in separate sections, we are now able to distribute the code to multiple nodes. A monolithic application is still possible, but in that case only a “migration” of the whole application from one node to another is feasible.

To accommodate this metamodel, a state and time models have been designed that allow to unambiguously determine data dependencies in the more general case of directed cyclic graphs. This feature is necessary to perform a dataflow-like execution of the application, but it is also useful during the design phase to observe the application evolution. In particular, it is feasible to “rewind” the execution to any previous state and change blocks on demand, to observe the impact of improved/debugged portions of code on partial execution. While these aids do not concern the execution in the “production” phase, they aim to provide more guarantees related to the correctness of execution by enhancing the “development” phase.

The application graph is, in general, a cyclic graph. This generality is necessary to account for complex iterative subroutines that are common, for example, within algorithms based on optimization. For this reason, the runtime is designed to address cyclicity by enriching data with temporal properties. These temporal properties allow for unambiguously determining the order of data even when code portions are run asynchronously, as it is the case in a distributed computation scenario.

Another important aspect of the metamodel is that all data types and code portions are hosted at external repositories whose authenticity can be verified by the wnode that requests the sub graph of the application to execute. In this way, a secure deployment is guaranteed as long as the repositories are not breached. The use of common repositories also enables a safe collaborative development, since data type specification is defined globally, possibly with explicit semantics defined using a natural language.

3.2.3 Development of the prototype

It must be remarked that the development status has not yet reached a demonstrable level of prototype. While the application metamodel has been defined fully, the software framework is still under preliminary development. The framework is written for the Java SE platform, targeting a Linux operating system. It must be noted that, while Atta itself is written in Java, the applications can be written in a mixture of different languages that are “glued together” using a data interchange format.

The setup part of the runtime is the one that received the great majority of implementation effort as of now, but it is clearly insufficient for a proper demonstration. The core of the framework is still the processing part, which from the point of view of a wnode involves the continuous retrieval of data for any code section to execute, and the notification of new data produced by the code itself.

The target embedded node for the prototype is the BeagleBoard XM, due to the requirement of reasonable processing and memory capabilities. For such nodes, the most sensible role is that of the wnode or snode, while the pnode, dnode and rnode roles are better suited to a (clustered) server. A cnode is simply a PC that controls/monitors the execution of an application.

3.2.4 Demonstration Scenarios

The applicability of the framework is clearly general, thus applying to all scenarios. However, the effort involved in the redefinition of the algorithms under the model-based paradigm suggests focusing on one particular scenario. The Vocal/Facial Recognition scenario is the preferable one due to the intensive workload involved and the spatially distributed nature of the data sources.

In this scenario, we expect to feature a set of embedded nodes to be used as wnodes. Each wnode will be able to access a video camera, in order to offer a multi-camera surveillance service. At least one PC node will be present to cover the other roles in the framework.

As remarked earlier, the Atta prototype brings the ability to easily deploy a new version of the Voice/Face Recognition algorithm. In addition, it will handle the possibility of node failures, where some video/audio sources become unavailable.

4 Reputation-based resource management technologies

4.1 Reputation and Trust

4.1.1 Definitions

Network security is a top priority requirement in the global environment of convergent internetworks, where information is conveyed interleaved and seamlessly. *Trustworthiness* is an important value when a network device or component needs to decide upon the next hop or path of transmission. It is especially important inside the intrinsic vulnerability of networking heterogeneous and wireless components, reflected in the evolvement of a series of security threats. A *trust management system* is a methodology to confront these attacks and a *trust model* is the method of estimating and establishing trust relationships among network nodes. While network availability, data integrity and confidentiality can be achieved with security methods based on key distribution, trust models are used for higher level security, stemming from needs oriented more to connectivity and routing. In its simplistic form a trust model functions in the direction of allowing a node to select the next hop based also in its recorded reliability. Of course, depending on the network layer implementation (routing protocol, etc.) other factors coexist and play their role (distance from destination, etc.) and their combination with the trust scheme is designed upon application needs. A few key notions follow:

- *Trust* is defined as the level of confidence of a component on another component, based on the perspective that the other component will perform a particular action important to the network, such as correctly forward a packet.
- *Reputation* is the opinion that one network node holds for another node.
- *Direct Trust* is the level of trust that a component forms about another component based on its own observations of the estimated node's behaviour in the network.
- *Indirect Trust* is the level of trust that a component forms about another component based on the concentration of observations made by third nodes and forwarded to it. Reputation and Indirect Trust can be used in an interchangeable manner.
- *Trust Value,* $T^{A,B}$, of node A for B, is the single value representing trustworthiness of B, calculated in any of the alternative trust model methods. Every node can preserve a *Trust Table* listing the trust values of other nodes of interest (e.g. neighbours). These values are updated periodically.

Trust models are more necessary in applications (or at these parts of the network), which do not rely on fixed reference points and the infrastructure is rather unstable, such as in ad-hoc and peer to peer sensor networks. These topologies, engaging wireless sensor networks are highly use case specific and the trust model design must be made taking into account their particularities. That is, the parameters that render a trust estimation scheme useful are the same that make it complex in its applicability and have to do mainly with the resource constrained environment of WSNs. Nodes may have limited memory, CPU, communication bandwidth and energy capacity. Therefore, the exact implementation of a trust model depends on the network characteristics, which affect the weight selection that different reputation parameters will be given. Smart solutions that can economize resources without affecting the effectiveness of the trust model can be adopted also. For example, the indirect trust values can be incorporated in beacon type messages, avoiding extra messaging and data overflow. Below, three examples of the flexibility in the parameterization of trust calculation are illustrated:

<u>Direct Vs. Indirect Trust</u>

When a node uses a combination of its own measurements and the indirect values to estimate another node's trustworthiness, the scheme is reputation based. The reason for the invocation of reputation values may be the inefficiency of direct measurements in the process of exacting a reliable trust value for each node of interest. Complementing direct values with indirect ones enhances the model's robustness and increases the network's protection. It comes though with the cost of increasing the consumption of resources and may degrade network's performance, as it requires extra resources for transmission, storage and processing.

Selecting the Metrics

A variety of observations (mirrored to metrics) can be used during the process of a direct trust value calculation. Therefore, the trust model designer has the convenience to highlight or suppress the contribution of some of the parameters, depending on the requirements imposed by the network. For example, since in WSNs energy is a critical resource, a metric considering energy of the next hop candidate can be incorporated in the trust model. A bigger weight coefficient can be given to this metric, to enhance the contribution of energy capacity in the decision about a node's trustworthiness.

Hierarchy and structure

Given the topology and network elements, one of the design's features is "where" (on which components) the trust model will be implemented. Several approaches exist, reflecting the specifications imposed by the application's architecture:

- Centralized: is efficient in cluster architectures. It's drawback is the vulnerability to attacks, since it is enough for the lead node to be compromised for the network to be seized
- Hierarchical: is suitable for networks that can be subdivided in islands (e.g. providing the same service) and its appliance reserves energy and bandwidth
- Distributed: several trust management schemes have been developed for the case where all nodes participate in the formation of trust tables. The advantage is the increased reliability of the model, whereas special care should be given on how to handle information flooding and overhead and the subsequent allocation of resources
- Hybrid schemes: combination of the above, depending on the partial characteristics of the network

It is worth noting that most of these models are interdependent with the management and knowledge of node's geographical information.

4.1.2 Trust related attacks

The trust management schemes are destined to confront security threats and enhance system's protection. A common attack is, for example, *Denial of Service (DoS),* consisted from a malicious node preventing nodes from using network resources (e.g. other nodes for routing purposes). The attack can be conducted in many forms. An effective trust model locates and isolates malicious nodes and thus eliminates the probability of a successful DoS attack. That is the reason that trust models are subjected to security attacks themselves. These attacks can be addressed by different security measures (e.g. authentication mechanisms), but should be taken into account during the trust model design phase, in order to enhance it (the trust model) with the appropriate defence measures (if the respective cost of resources is affordable). Among the attacks threatening the correct trust table calculation are:

Bad mouthing attack

An attacker node propagates false reputation values for malicious nodes, in order to increase their trust values.

Conflicting behaviour attack

A malicious node can follow a contradictory behaviour pattern towards two nodes. This will probably result in the two nodes estimating a low trust value for each other, whereas there is no reason to do so.

Sybil attack

The attacker is forging multiple identities, increasing its probability of being chosen as the next hop.

On-Off attack

A malicious node behaves successively bad or well, aiming to confuse the trust calculation scheme.

Newcomer attack

A malicious node rids its bad history by registering as a new user.

4.2 Reputation based Secure Routing

4.2.1 Description

In distributed systems, each entity must depend on its neighbours to carry out a transaction and accomplish full communication among all participants. Routing protocols are implemented for this purpose [3]. Their basic functions are routing and forwarding. Routing is the process of establishing a communication path between two end nodes. Forwarding is the transmission of the traffic through the selected path. Most routing protocols base the routing process on distance metrics among a path's nodes. The protocol selects the shortest path.

Reputation-based schemes are used in wireless networking to provide secure routing functionality [4]. Due to the open medium and the dynamic entrance of new nodes to such networks there must be a way to establish trust relationships to avoid malicious entities. Reputation is formed by a node's past behaviour and reveals its cooperativeness. A node with high reputation can be considered as trustworthy. Legitimate nodes depend mostly on trustworthy entities to accomplish communication tasks, like routing and forwarding. Also low reputation can reveal selfish or malicious entities and is used for intrusion detection. Legitimate nodes try to avoid such entities and not serve their traffic.

A common approach for implementing secure routing functionality is the integration of a routing protocol with a reputation scheme. Reputation and trust information are included in the decision making process along with the distance metrics. For the routing process, the goal is the selection of short paths with well-reputed nodes. Thus, legitimate entities avoid malicious ones. For the forwarding process, the goal is to serve only legitimate entities and isolate the malicious or selfish ones.

Many reputation-based schemes for secure routing have been proposed and each one provides protection against a set of security attacks and vulnerabilities. The schemes embody features to form reputation and trust. These features add complexity and process overhead to the pure routing protocol. The basic trade-off that is encountered is between security and performance. As security goes high, the overhead to support the level of security goes also high and the system becomes more complex. The selection of the proper scheme depends on the application properties. Networks with ultra-constraint devices cannot support heavy reputation-based schemes that offer high levels of security. A network manager has to apply one of the proposed implementations without having the ability to adapt the scheme to its application's needs.

4.2.2 Routing protocols

The design and adoption of a suitable routing protocol is a task affiliated with many parameters, especially in the existence of environment sensitivities (wireless links) and device constraints (nodes with limited resources). Some of the dimensions taken into account are node mobility, deployment possibilities, coverage requirements, heterogeneity, topology, architecture, infrastructure, and energy constraints. Therefore, in the context of wireless sensor networks, which is the main framework of the proposed reputation based management resources scheme, the security and geographical implications (to name two factors) impose the non-exhaustive categorization described in the following paragraph.

4.2.2.1 Taxonomy

A common method of categorization is the one initiating from the application's topology implications. The components may not be fully aware of the system's infrastructure and therefore not aware of the available routes. This is a kind of information that has to be retrieved and there are two methods of doing so: proactive and reactive one.

<u>Proactive routing</u>

The routing protocols are table-driven, maintaining routing tables updated in regular intervals

<u>Reactive routing</u>

On the contrary with proactive routing, protocols find routes on demand.

Pros and cons of the two methods have to do with trade-offs between network latency, time of network adaptation, performance and data overhead.

However, probably among the most frequent classifications is the one that considers the communication network structure discriminating protocols in three types: data-centric, hierarchical and location-based. Other classification methodologies exist, however this is the discrimination which will be analysed for the selection of a routing protocol, which in turn will be the carrier of the reputation scheme.

Data centric

The routing process is based on the rationale of selecting a set of nodes to direct traffic, in cases where network size renders inefficient to assign unique identifications to all nodes. Subsequently, a complementary methodology of resources management and selection of next traffic hop regions is needed. Otherwise the overhead is not affordable and the scheme is not cost efficient in terms of bandwidth and energy. Several protocols and mechanisms can be found in the bibliography.

Hierarchical

Protocols of hierarchical routing focus on limiting locally sensors' use of resources and network overhead, with the formation of clusters, where information will be fused, processed and selectively forwarded. The optimization objective here is the approach on forming these node groups.

The well know protocol LEACH (Low Energy Adaptive Clustering Hierarchy), fall under this category. Each node uses a stochastic process to determine whether it will become a cluster head in this round. LEACH assumes that although a node has a radio powerful enough to directly reach the base station, using this radio at full power all the time would exaggerate energy limits. This means that randomly selected cluster heads are regularly changed to avoid energy consumption of these specific nodes. Data processing is performed inside each cluster and the output is channelled to sink nodes. About 5% of all nodes are clusters each at each given time. LEACH is a MAC protocol, where sensors are communicating with their cluster head through TDMA at timeslots allocated by the cluster head. CDMA is also used, to minimize interference between clusters. The protocol is suitable for application where continuous network control and monitoring is required.

Location-based

As denoted by its name, protocols of this routing mechanism form their route selections based on nodes' geographical information. In the absence of a positioning system (e.g. GPS) indirect localization methods are employed, such as estimation based on signal strengths or exploitation of neighbouring nodes known positions. Distance and relative geographical information are used to avoid useless transmissions. Given that the challenges of location verification are addressed successfully, location-based routing is appropriate for networks with sensor mobility. The reputation scheme proposed in this chapter will be based on geographical routing.

4.2.2.2 Comparison

Communication processes, in the context of an efficient resource management scheme, require distributed routing methods, with the cooperation of adequate number of nodes. Sometimes the system's architecture is represented by an unstable or changing infrastructure. In the following we stress that geographical routing could be a solution applied in nSHIELD subsystems, to provide a trusted connectivity scheme, since it combines and supports security, node mobility (Railway Scenario in our case), scalability, composability and the desired SPD level.

The core idea resides in the concept to use localization techniques and information instead of measuring hops and flooding network nodes, in order to create routing tables and maps. This means, in its simplistic form and in absence of other criteria, that each time a node needs to send a packet, it will forward it to the neighbouring node closer to the destination, based on the aforementioned geographical recent history data. The information is restricted for one-hop neighbours for each node, since with this information and the destination address the one-hop neighbour closest to destination is selected. This reduces unnecessary overhead, stemming from economy in a series of parameters:

- Routing tables contain only one-hop neighbours
- Location updates can be incorporated with – by default and periodically transmitted – beacons
- The network is scalable, since the one hop interest is indifferent of additions
- Nodes can be mobile and this will be reflected in beacons

- Some categories of attacks are neutralized (e.g. sybil, selective forwarding), due to their nature relating to the node ID or location

The following paragraph examines the aspects of insertion of trust metrics in a geographical routing algorithm, for the construction of a reputation based routing algorithm, before proceeding to its implementation.

4.2.2.3 Combining Reputation and Routing

Three options to combine reputation and routing exist. Their applicability depends on the priority the user wishes to grant on a service or resource (trust vs. distance):

a) Trust weighted next hop selection:
   1. Define trusted neighbour set (TNS), which includes the k most trusted neighbours
   2. From nodes in TNS, choose the node located closest to the destination

b) Distance weighted next hop selection:
   1. Choose the k nodes closest to the destination
   2. From these nodes, select the most trusted

c) Trust weighted next hop selection (threshold):
   1. define the TNS using an application specific trust threshold
   2. from nodes in TNS, choose the node located closest to the destination

4.3 nSHIELD Reputation scheme

We introduce a Secure Routing Framework, based on 11 components. Also, a Trusted Routing scheme, implementing Direct and Indirect Trust over a geographical routing protocol (Greedy Perimeter Stateless Routing, GPSR) and an Intrusion Detection System (IDS) implemented on a wireless sensor node distributed architecture. Direct Trust scheme is the simplest form of Trust establishment, as it takes into account only first-hand observations of the cooperative interactions (e.g. packet forwarding for secure routing), which can lead to increased detection time of attacks. Indirect Trust scheme adds Reputation calculations from other nodes, which can lead to a more complete view of node behaviour and shorten the time needed to learn a node's trustworthiness compared to the scheme implementing only Direct Trust. Reputation based IDS employing a Bayesian formulation, specifically a Beta reputation system for reputation representation can be considered the solution providing the highest SPD level, using only third party information of high integrity and broadcasting alerts, when non-trustworthy behaviour is detected. The three achieved SPD levels are presented in the table below.

Table 4-1: Achieved SPD levels (reputation scheme)

SPD Level	Nano
1 (lowest)	-
2 (low)	DT (Direct Trust), A4
3 (medium)	Weighted DT (Direct Trust) + ID (Indirect Trust), A4 + A5
4 (high)	Reputation based IDS algorithm, A6 + A7

4.3.1 Trusted Routing Framework

For nSHIELD network layer, we implement a novel module reputation-based scheme that can act as a general purpose scheme for a wide range of applications. The basic idea is to identify the common components of reputation-based schemes and provide an abstract framework. We identify eleven components where each one of them serves a specific functionality. For every component we propose a set of features that implements the component's functionality. The segmentation of the scheme into components enables the dynamic deployment and extension of the scheme. As new features and trends are proposed in the field of reputation-based schemes, we can simple add these features in the components container.

The network manager selects which components are active and the exact set of features that implements them. During the selection process, a designer could model the combination of more than one feature for some components. The designing options can range from ultra-lightweight schemes to heavily secure ones. The selection decision will be either static at deployment time or dynamic at run time, if such operation is supported. Also, heterogeneous nodes could utilize different features for some components.

The 11 components are:

1. **Knowledge type:** *Direct* knowledge is the direct opinion that an entity possesses about other entities and is determined by their previous transactions or observations of certain factors. *Indirect* knowledge is the opinion that other entities possess about the investigated entity.
2. **Transaction evaluation:** defines the result of a transaction and decides what will happen next. *Simple* evaluation denotes the transaction result (success or fail) and proceeds to the next step. *Congestion windows* and *communication channel observation* can be used as a tolerance mechanism if failures occur during periods of traffic congestion and bad channel conditions respectively. Another option is the *re-routing* of a failed transaction.
3. **Transaction grading:** defines the exact value that is applied for the examined transaction, after the transaction evaluation component execution. *Simple* and *gradual* grading is implemented.
4. **Reputation evaluation scope:** The evaluation scope indicates which entities of the network are about to be examined by the reputation scheme. Three categories are considered: *node*, *path* and *community* of nodes.
5. **Reputation calculation:** determines the current reputation value of the examined entity. Four formulas are supported. The *simple summation* is the summation of the transaction grading values. The *reputation fading* stores a small history of grading values. The values are weighted according to time and reputation fades – indicating that most recent values are considered more important. The *reputation normalization* defines a statistical normalization of the reputation history, where the extreme values are ignored. The *fading of the normalized reputation* combines the two previous formulas. The reputation history is statistically normalized and then the fading formula is calculated.
6. **Notification strategy:** if indirect knowledge is supported, a node can send and receive messages with indirect knowledge. Three types of notifications are implemented. The *negative* notification signalizes a misbehaving node. Negative notifications are usually exploited by badmouthing attacks. *Positive* notifications signalize nodes that behave according to the rules. Both *positive and negative* notifications are used to signalize both misbehaving and well-behaving nodes.
7. **Notification scope:** the group of nodes that will receive a notification. Three groups are considered: *broadcast*, *trusted/friends*, *misbehaving node*. If the broadcast scope is selected, the node will broadcast its notifications to all its neighbours. If the trusted/friends scope is selected, the node will send its notifications only to its trusted/friend nodes. If the third scope is selected, the node will send a warning message to the misbehaving node. If the examined node ignores the warnings and continues misbehaving, it is categorized as malicious.
8. **Indirect trust evaluation:** If indirect knowledge is supported, the node needs to evaluate the notifications it receives. Three formulas are supported. With *simple* evaluation, the node processes all pieces of indirect knowledge the same. With *deviation test*, the node checks if the indirect knowledge it receives, deviates significantly from its direct knowledge. With the third option, the indirect knowledge is *weighted*. The notifications sent by trusted nodes, gain higher weight.
9. **Punishment strategy:** defines the thresholds for marking suspicious and malicious nodes as well as the type of punishment. Punishment types include the discarding from the *routing* process, the termination of packet *forwarding* for the punished nodes and the combination of both punishments for *routing and forwarding*. If *no* punishment is selected, nodes are ranked without penalizing their

behaviour. Paths with misbehaving nodes produce low reputation and are more difficult to be selected by the path selection component (see below).

10. **Initialization and re-entrance strategy:** Initialization defines the default reputation values for a new entity. Re-entrance strategy allows a punished node to re-enter the network with a default reputation value under some conditions. The strategies that are currently supported are the *periodic* re-entrance, the *redemption* and *no* re-entrance. With the periodic option, a punished node re-enters the network after 'Pt' minutes. With redemption, all bad ratings are recalled to neutral ones after 'Rt' minutes. If no re-entrance is allowed, a punished node can't re-enter the network.
11. **Path selection:** indicates the criteria for deciding which path to choose during the routing process. We support the *shortest* path, the most *well-reputed* path and the *shortest well-reputed* path.

The most important operation of a reputation-based scheme is the calculation of reputation and trust. A node continuously receives new pieces of knowledge both from its direct interaction with its neighbours and the notifications from other nodes. There are two evaluation operations for direct and indirect knowledge respectively. When new knowledge has been evaluated, the reputation and trust values are updated. If the trust level of the node has changed, notifications can be sent.

```
Algorithm 5 Reputation & trust evaluation – Receive new knowledge
if (the new knowledge is DIRECT knowledge) then
   (Evaluate the transaction result)
   (Grade the transaction according to the direct knowledge scope)
   (Calculate the new reputation and direct trust values for this scope)
end
else if (the new knowledge is INDIRECT knowledge) then
   (Evaluate the new piece of knowledge according to the indirect trust
   evaluation strategy)
   (Calculate the new indirect trust value for the indirect knowledge scope)
End

Update the punished nodes list
Update the total trust value for this scope (direct or indirect)

if (this node sends indirect knowledge to others AND the status for the
punished nodes changes) then
   (Send indirect knowledge to other nodes according to the Notification
   Strategy)
end
```

Algorithm 5: Reputation & trust evaluation – Receive new knowledge

The following figure illustrates the process of evaluating new knowledge.

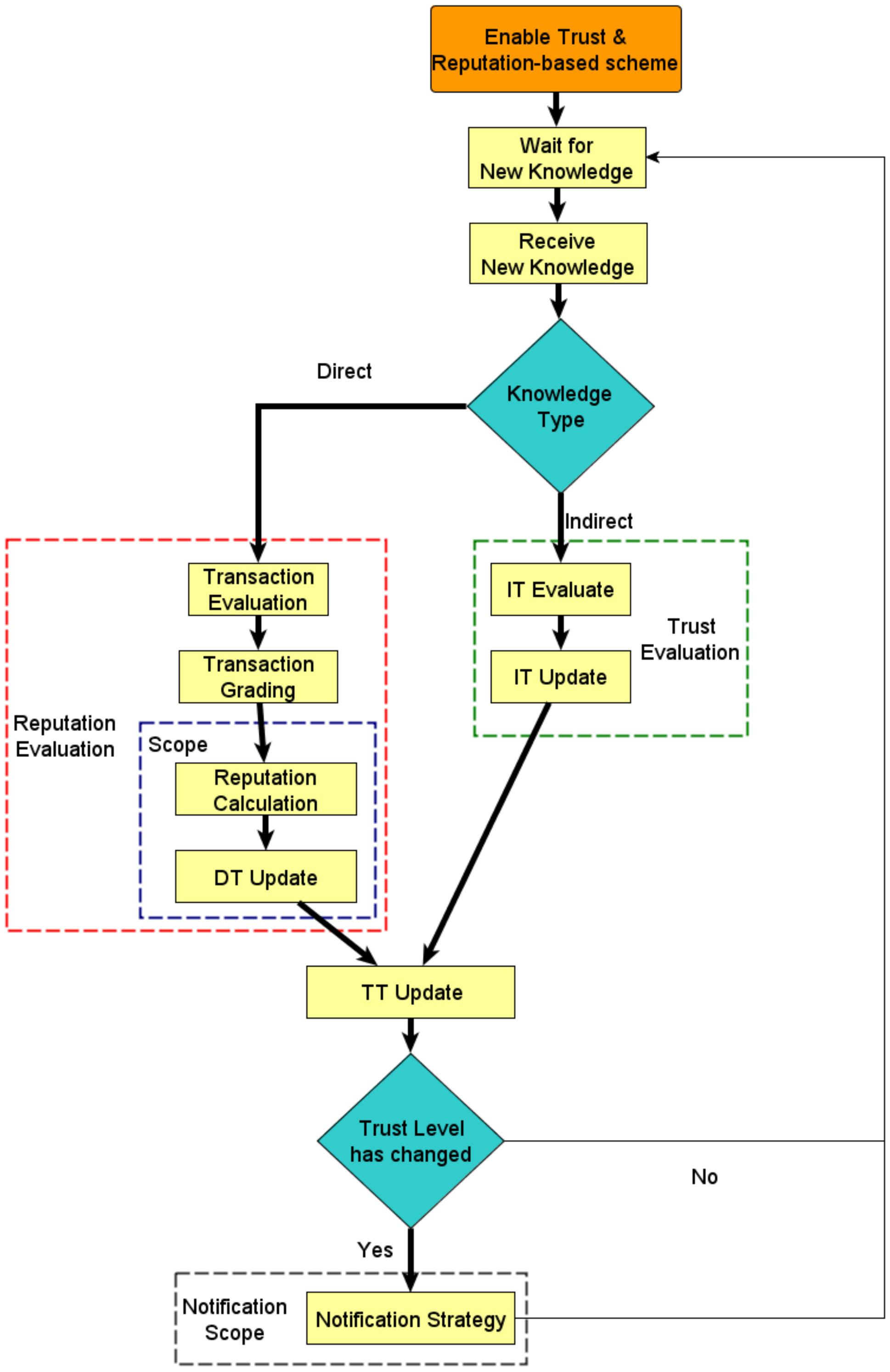


Figure 4-1: Control flaw diagram of the proposed reputation-based scheme

Other important operations are the path selection and the entrance of new or previously punished nodes. When a new communication path is established, the scheme has to decide which nodes will be included. Through the path selection component we can denote the selection strategy of the scheme. Also, if the re-entrance strategy is enabled, it is periodically examined the case of permitting to punished nodes to re-enter the network.

We implement a GUI to ease the framework's configuration. All the aforementioned component and parameters are described and the user can select the most appropriate for his network. Before the configuration process is completed, all the parameters are presented to the user for a final confirmation. To make our proposal more applicable and acceptable, we have pre-set the configuration options for implementing the decision making process of well-known reputation and trust schemes for secure routing. These schemes are the Watchdog and Pathrater [5], CONFIDANT [6], Improved CONFIDANT [7], CORE [8], Reputated-ARAN [9], CSRAN [10], RFSN [11] and a Semi-distributed reputation-based Intrusion Detection System for mobile Ad-hoc Networks [12].

Watchdog and Pathrater [5] are the main components of a reputation scheme for secure routing. Watchdog eavesdrops the wireless channel and identifies the misbehaving nodes that don't properly re-transmit a packet as "next-hops". Based on these observations, Pathrater ranks the nodes and find routes that don't contain these nodes. However, it doesn't punish misbehaving nodes and they are encouraged to continue misbehaving. Also, it doesn't utilize indirect knowledge and can't work properly in the presence of collisions. The scheme is the most lightweight for secure routing and can be applied to Nano nodes.

CONFIDANT (Cooperation Of Nodes, Fairness In Dynamic Ad-hoc NeTworks) [6] extends the functionality of Watchdog and Pathrater by punishing misbehaving nodes. The punished nodes are excluded from routing and their packets aren't forwarding by the legitimate nodes. Moreover, the scheme uses negative warning messages to inform about misbehaving nodes. However, the negative warnings may be exploited by badmouthing attacks. CONFIDANT can also be applied to Nano nodes.

Improved CONFIDANT or Robust Reputation System (RRS) [7] upgrades the original proposal. It utilizes reputation fading, negative and positive notifications as well as reputation and trust metrics. When indirect knowledge is received the node performs a deviation test and ranks both the reported node and reporting one. The scheme effectively forces a malicious node to continually cooperate with legitimate nodes to survive in the network. The Improved CONFIDANT is more resource demanding than the original version and is suitable for micro/personal nodes.

CORE (Collaborative REputation) [8] is another lightweight reputation scheme. It uses both direct and indirect knowledge and detects a specific type of misbehaving nodes, the selfish ones. The forwarding operation is considered more valuable for the network's performance than routing and gains higher weight in reputation's calculation. In contrast with CONFIDANT, it uses positive reports to prevent false accusations. Negative rating is given only by direct observations. If a misbehaving node is detected, it will be rejected by the network. However, CORE doesn't detect malicious nodes. It is suitable for Nano nodes.

Authenticated Routing for Ad-hoc Networks (ARAN) is a routing protocol for ad-hoc networks that uses certificates to authenticate each node. The protocol detects most of the attacks performed by malicious nodes, such as Sybil and newcomer attacks. The reputation schemes Reputated-ARAN and CSRAN extend the routing protocol to detect attacks that can be performed by authenticated nodes, especially selfish ones. **Reputated ARAN** [9] uses simple reputation metrics with incremental increase and gradual decrease and no indirect knowledge. **CSRAN** [10] uses both trust and reputation. For the reputation calculation utilizes a Bayesian procedure to perform reputation fading. Furthermore, when a node infers that the next hop in a route is malicious and hasn't forwarded the communication packets it will automatically re-route the communication from that point to the destination. Thus, the network's performance is improved as fewer failures occur. However, the re-routing process could be exploited by malicious nodes to re-route traffic to specific nodes, either malicious or legitimate ones. As the ARAN protocol uses certificates it is developed for devices with sufficient computational capabilities. Reputated-ARAN and CSRAN are applicable to micro/personal and power nodes. Our proposed reputation framework simulates the decision making process for these two protocols in the applied routing protocol.

RFSN (Reputation-based Framework for Sensor Networks) [11] uses trust and reputation. Based on reputation, nodes form communities of trust. Reputation fading is performed with a Bayesian formulation. The scheme is implemented as a middleware application in WSNs and was applied on MICA2 motes running TinyOS and SOS. The scheme is suitable for micro/personal nodes.

Semi-distributed reputation-based Intrusion Detection System for mobile Ad-hoc Networks [12] proposed implements many novel reputation metrics and provides protection against a wide range of attacks in MANETS. It uses direct and indirect knowledge, reputation fading, and redemption for automatically re-entering and is tolerant in failures due to traffic congestion. The scheme gives more weight to recent and direct knowledge and applies incremental increase and gradual decrease. Both negative warnings and avoid lists are used as indirect knowledge. The nodes are categorized as normal, suspicious and malicious. A node is recognized as malicious only by direct observation. The scheme is suitable for micro/personal and power nodes.

The network designer is able to add functionality, increase the level of security and adopt the final scheme to his needs. Under appropriate configuration, the proposed scheme can detect and counter the following types of problematic conditions:

- **Suspicious** behaviour due to:
    - Bad communication channel conditions (e.g. jamming attacks)
    - Traffic congestion
- **Selfish** behaviour of nodes that:
    - Don't forward packets that belong to other nodes
    - Only demand from others to server their own packets
- **Malicious** behaviour where a node:
    - Always performs malicious activities
    - Gains high reputation to perform effective attacks later
    - Performs jamming attacks
    - Tries to decrease the reputation value of legitimate nodes
    - Tries to increase the reputation value of other malicious nodes
    - Wants to find out the nodes that are able to detect its malicious behaviour

Moreover, the configuration parameters can be altered at runtime. Consider a scenario where we have set the Watchdog and Pathrater scheme in a WSN with nSHIELD Nano nodes. Then, the overlay layer becomes aware of an emergence situation and informs the overlay security agents to increase the security level of the underling networks. The security agent, who manages the examined WSN, checks its policy and informs the sensor nodes to increase their security. The policy orders a specific set of actions, like lowering the threshold for malicious nodes and applying the routing and forwarding punishment strategy. The WSN is conformed to the new policy, becomes stricter with misbehaving nodes and isolates the malicious ones. Similarly, when the emergence situation is over, the overlay security agents can set the underling networks back to their normal form.

4.3.2 Trusted GPSR

4.3.2.1 Direct Trust Metrics

In the trust management scheme a node updates its trust tables periodically, having a new trust value for each candidate next hop neighbour. With regards to the direct component of this calculation, there is a series of observations on which the node can build a series of metrics, possibly heterogeneously weighted. The sum of these weighted metrics (or the weighted sum of metrics, depending on the followed method), provide the direct trust value of our node for every one of its neighbours. It is self-evident that this value will eventually depend on the general node behaviour in network's functionality. A tentative non-exhaustive list of direct trust calculation events is the following:

Packet forwarding

The node preserves a counter of the packets successfully (or not) forwarded by the nodes belonging to the trust table list. Packets can concern data or control information or both, assigning corresponding weights or not.

Packet Integrity

The node preserves a counter of the packets successfully and unaltered forwarded by the nodes belonging to the trust table list. Again, the metric can concern data as well as control packets.

Authentication

The node uses a mechanism to authenticate the packet of a neighbouring node and increase or decrease its trust value accordingly.

Cryptography

The trust value of an observed node is increased when it uses a cryptography-confidentiality mechanism.

Response to Reputation

When a node is cooperative in the reputation requests its trust value increases in the trust table of the requestor.

Energy

The level of remaining energy can be considered as metric to protect trusted nodes from exhaustion. The extent of this metric is up to the designer (the threshold is flexible).

History

The node preserves logs of the cooperation messaging with other nodes. This measure mitigates On-Off attacks (mentioned above).

4.3.2.2 Evaluation of Trust

The reputation information collected by a node during the stage of indirect trust calculation can be requested reactively. However, in order to avoid excessive transmission the designer can resort to alternative solutions, such as include reputation data in messages such as BEACON or HELLO. Beacon messages are automatically sent by sensors periodically to denote their ID and possibly location, depending on the implementation of routing protocol.

Below, is depicted as an example, a rough mathematical outline for the calculation of a node's B total trust value, in the trust table of a node A. It is calculated as a combination of Direct Trust values and Reputation.

Firstly the direct trust value corresponding to the observation of an event of type *i* is:

$$T_i^{A,B} = \frac{a_i S_i^{A,B} - b_i F_i^{A,B}}{a_i S_i^{A,B} + b_i F_i^{A,B}} \qquad (1)$$

Where

- $S_i^{A,B}$ is the number of successful type *i* events that A has measured for B
- $F_i^{A,B}$ is the number of failed type *i* events that A has measured for B

a_i and b_i represent the weight of a success vs. the weight of a failure of type E_i events. The above equation for a_i=b_i=1 becomes similar to the one presented in [13], while changing the relation between a_i and b_i we can give more importance to the positive (successful) or the negative (failure) events.

If we assume *k* types of observation events, the total direct trust value of A for B is:

$$DT^{A,B} = C^{A,B} * (\sum_{i=1}^{k} W_i * T_i^{A,B}) \qquad (2)$$

Where:

- W_i is the weighting factor for each one of the *k* event types
- $T_i^{A,B}$ is node's A trust value of event *i* regarding node B
- $C^{A,B}$ is the confidence factor derived from the equation

$$C^{A,B} = 1 - \frac{1}{noi + a} \qquad (3)$$

Where:

- noi is the number of interactions
- a is a positive integer factor

As regards to the indirect component, it is constructed from the direct trust values. Specifically:

$$IT^{A,B} = \sum_{j=1}^{n} W(DT^{A,N_j}) * DT^{N_j,B} \qquad (4)$$

Where:

- *n* is the number of neighbouring nodes to A
- N_j are the neighbouring nodes to A
- $DT^{N_j,B}$ is node's N_j reputation value of node B
- $W(DT^{A,N_j})$ is a weighting factor reflecting node's A direct trust value of node N_j

Finally the total Trust Value of A for B is:

$$TT^{A,B} = W(DT^{A,B}) * DT^{A,B} + W(IT^{A,B}) * IT^{A,B} \qquad (5)$$

Where

- $DT^{A,B}$ is node's A direct trust value of node *B*
- $W(DT^{A,B})$ is a weighting factor reflecting node's A direct trust value of node *B*
- $IT^{A,B}$ is node's A indirect trust value of node *B*
- $W(IT^{A,B})$ is a weighting factor reflecting node's A indirect trust value of node *B*

All the weights are subject to modifications, per application case, to reflect the nodes' and network status.

4.3.2.3 Implementation

For the nSHIELD trusted routing operation calculation of direct, indirect and total trust calculation using pseudo-code presented in Algorithm 6 and Algorithm 7 are a first prerequisite to extract the trust value based on which routing decisions will be taken. We propose the adoption of GPSR (Greedy Perimeter Stateless Routing) [14] as the routing protocol, upon which the reputation scheme will be implemented. The reasons for this selection are the advantages that GPSR comes with which include guaranteed delivery, low algorithmic complexity and routing information storage, ability to support large scale networks and adaptability in network changes as beacon messages proactively inform nodes about neighbours' changes and new additions.

```
Algorithm 6 Direct Trust Calculation
Select trust metrics and set a weighting factor for each trust metric
for (each interaction)
   (Check node ID of the incoming interaction)
   if (new node ID) then
      (Store neighbouring node ID in Direct Trust Table)
      (Set number of interactions to zero for new nodes)
      (Set initial trust value for the node)
   else
      (Update number of interactions )
   end if

   if (cooperative interaction) then
      (increase successful interactions)
   else
      (increase failed interactions)
   end if
   (Calculate Direct Trust using equation (2) and store new value)
end for
```

Algorithm 6: Direct trust calculation for trust establishment

```
Algorithm 7 Indirect and Total Trust Calculation
Step 1: Broadcast of First-hand evidence
if (Time equal to periodic trust transmission interval) then
   (Create packet of first-hand observations)
   (Send first-hand observations table to all neighbouring nodes)
end if

Step 2: Receipt of First-hand evidence
(Set initial weighting factors for indirect trust)
if (reputation packet received) then
   for ( each node ID contained in reputation table)
      if (sender node is confidant) then
         (Calculate indirect trust using equation (4))
         (Calculate total trust using equation (5))
      else
         (Decrease weighting factor of the sender)
      end if
   end for
end if
```

Algorithm 7: Total trust calculation

4.3.3 Intrusion Detection in Wireless Sensor Networks

As discussed in [15], different versions of IDS have been proposed in recent years with multiple architectures for detecting WSN attacks. We consider IDS based on cooperation between nodes and fully distributed architecture. Keeping these two main features we propose architecture of cooperation, based on reputation to create a network of autonomous sensors capable of detecting most kind of attacks and network failures using an anomaly detection system together with specification-based detection system. All this designed from the premise of creating a system that fits the characteristics of sensor networks and maintaining the protocol as lightweight as possible to guarantee the autonomy of the nodes.

4.3.3.1 Description

The IDS proposed is a distributed anomaly detection based system, where each node will have an IDS agent that will monitor local activities. If the local agent cannot determine the behaviour of an activity, this agent will interact with the agents near him to determine if that activity is malicious or not. Once that one activity is considered malicious, the IDS will take necessary measures to mitigate the situation.

IDS agents located in nodes are compounded by five parts; Local Data Collection module gathers different inputs, systems logs, network traffic or sensor values. After, recollected inputs are analysed by Local Detection Engine that will raise alarm flag if finds evidence of malicious activities. Once alarm is raised the Local Response and Global Response will take care of the situation to mitigate the failure. But if the Local Detection engine cannot determine the conduct of certain behaviour, the Cooperative Detection engine will use other nodes' opinion, to define if it is normal or malicious.

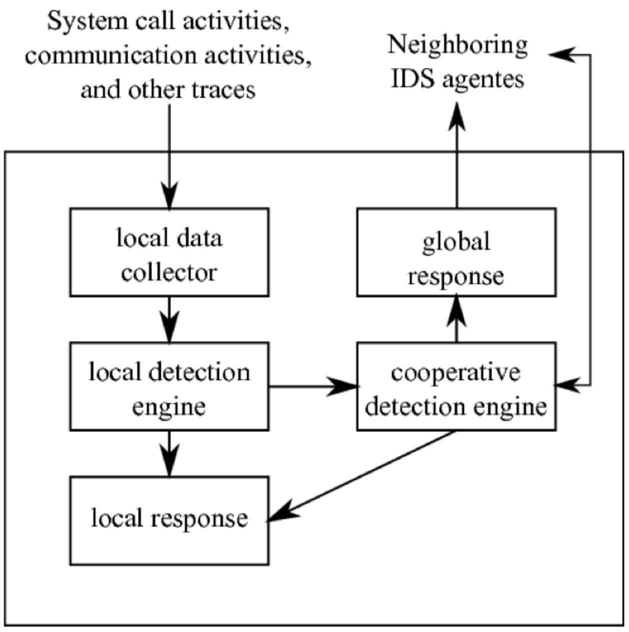

System call activities, communication activities, and other traces
Neighboring IDS agentes
local data collector
global response
local detection engine
cooperative detection engine
local response

Figure 4-2: Local IDS architecture

The detection system proposed for this IDS is a hybrid between anomaly and specification-based detection systems. At the initialization of the system, several specific parameters are configured, such as response time and frequency of notifications. This allows the IDS to monitor the different protocols of the system and search for possible attacks.

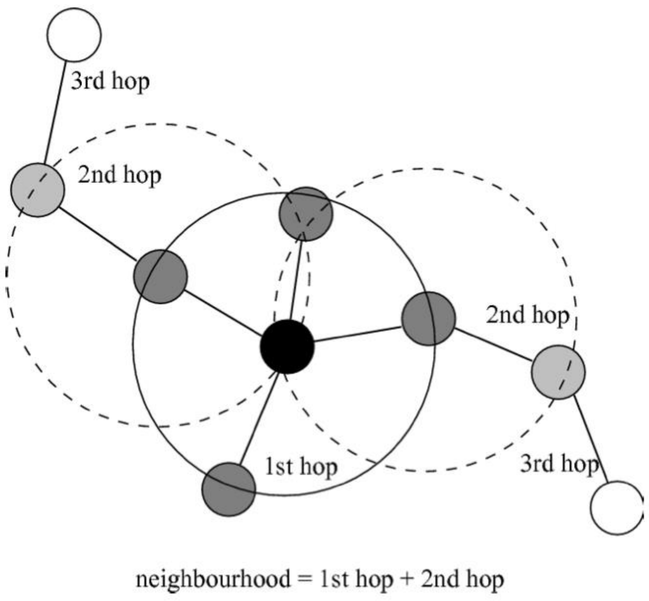


Figure 4-3: Node neighbourhood definition

To determine the behaviour of a node, reputation and trust are used. If node A suspects about the confidence of node B, node A can use other's nodes reputation value for node B. The most shared opinion about the confidence of node B can confirm or discard the suspicions of node A.

To determine the reputation of node B for node A, node A takes into account the communications and interactions between them. In order to estimate the reputation, the Beta distribution is used [16] [17], which uses cooperative (successful) and non-cooperative (failed) interactions to estimate the reputation value. The reputation of a node *j* for node *i* is defined as:

$$R_{ij} = \beta(\alpha_j + 1, \beta_j + 1)$$

Where:

- R_{ij} represents the reputation of *j* for node *i*,
- α_j are the cooperative (successful) interactions of node *i* with node *j*
- β_j are the non-cooperative (failed) interactions of node *i* with node *j*

We can also determine the trust on node *j* by node *i* by calculating the statistical expectation of the reputation distribution function, given by:

$$T_{ij} = \mathrm{E}(\beta\{\alpha_j + 1, \beta_j + 1\}) = \frac{\alpha_j + 1}{\alpha_j + \beta_j + 2}$$

Where:

- T_{ij} represents the trust on *j* by node *i*,
- α_j are the cooperative (successful) interactions of node *i* with node *j*
- β_j are the non-cooperative (failed) interactions of node *i* with node *j*

On the other hand, if a node has doubts about another node and cannot determine with certainty if it is malicious or not because of lack of data, this node can use information from its neighbourhood to determine with accuracy the aim of the suspicious node. The neighbourhood of a node is compounded by the nodes that are not farther than two hops.

As stated in [18], we would also like to point out that the objective of the integration step is not to establish consistent reputation metrics at all the nodes. Our only objective is to make nodes share experiences (via reputation). This can be conceptualized as external evidence for a node to establish more concrete reputation metrics. This is useful for several reasons:

- propagating reputation information in the network facilitates the formation of the community of trustworthy nodes, convergence time is shorter than when relying solely on direct observations,
- acquiring information through external evidence is much cheaper from an energy perspective, as you get information about several nodes in a single packet (observation), and this allows nodes to learn about distant nodes (not in their neighbourhood), which it cannot monitor directly.

To calculate the new reputation of a node using the information shared by the nodes in the neighbourhood, the same system of second hand information used in [18] and explained in [17] is applied. To determine the new α_j and β_j parameters of the Beta distribution for a given node *j*, when node *i* uses the information shared by node *k*, the following equations are used:

$$\alpha_j^{NEW} = \alpha_j + \frac{\{2 * \alpha_k * \alpha_j^{\,k}\}}{\{(\beta_k + 2) * (\alpha_j^{\,k} + \beta_j^{\,k} + 2)\} + \{2 * \alpha_k\}}$$

$$\beta_j^{NEW} = \beta_j + \frac{\{2 * \alpha_k * \beta_j^{\,k}\}}{\{(\beta_k + 2) * (\alpha_j^{\,k} + \beta_j^{\,k} + 2)\} + \{2 * \alpha_k\}}$$

Where:

- α_j^{NEW} is the new value for the cooperative interactions of node *j* with node *i*
- α_k are the cooperative interactions of node *k* with node *i*
- β_k are the non-cooperative interactions of node *k* with node *i*
- α_j^k are the cooperative interactions of node *j* with node *k*
- β_j^k are the non-cooperative interactions of node *j* with node *k*

With this second hand information a node can calculate the new interaction values that are used to recalculate the reputation. Moreover, information received from nodes with high reputation will have greater weight than those with less reputation (as those nodes with a higher reputation will have larger α_k values and smaller β_k values).

The extra advantage of incorporating indirect (second hand) reputation into the system comes with a cost: the system is more vulnerable to bad-mouthing attacks. Malicious nodes can propagate negative reputation ratings of other good nodes in order to lower their reputation values. This penalizes those good nodes when taking into account the reputation information they publish. This in turn means that even if those good nodes publish bad reputation information for the malicious nodes (which they should) the malicious nodes' reputations are almost not affected.

This is why nodes usually classify cooperating and non-cooperating nodes separately in their local tables. And they only share reputation information about cooperating nodes, thereby sharing only good reputation information and reducing this kind of attack. But this also means that it takes more time to effectively detect and punish malicious nodes.

In order to obtain some computational simplification for the implementation, we have decided to use a modification of the Bayesian method presented so far. This modification also addresses the self-imposed limitation described above (only publishing good reputation information). Also, as stated previously, second hand information from other nodes can be requested reactively. However, in order to avoid excessive transmission the designer can resort to include reputation data in messages such as BEACON or HELLO. In this case, we propose to also publish the first-hand information when a given node is considered malicious.

In this approach [19], a node *i* maintains two ratings about every other node *j* that it cares about. The Reputation rating represents the opinion formed by node *i* about node *j*'s *behaviour*. The Trust rating represents node *i*'s opinion about *how honest* node *j* is (note that this is slightly different from the definition we've used before for Trust).

We represent the ratings that node *i* has about node *j* as data structures R_{ij} for reputation and T_{ij} for trust. In addition, node *i* maintains a summary record of first-hand information about node *j* in a data structure called F_{ij} .

First, whenever node *i* makes a first-hand observation of node *j*'s behaviour, the first-hand information F_{ij} and the reputation rating R_{ij} are updated. Second, from time to time, nodes publish their first-hand information to their neighbours. Say that node *i* receives from node *k* some first-hand information F_{kj} about node *j*. If *k* is classified as confident ("trustworthy") by *i*, or if F_{kj} is close to R_{ij} (in the sense that is shown below) then F_{kj} is accepted by *i* and is used to slightly modify the rating R_{ij} . Else, the reputation rating is not updated.

In all cases, the trust rating T_{ik} is updated. If F_{kj} is close to R_{ij}, the trust rating T_{ik} slightly improves, else it slightly worsens. The updates are based on a modified Bayesian approach and on a linear model merging heuristic. Note that, with this method, only first-hand information F_{ij} is published. The reputation and trust ratings R_{ij} and T_{ij} are never disseminated. The ratings are used to make decisions about other nodes, which is the ultimate goal of the entire reputation system.

The first-hand information record F_{ij} has the form (α, β). It represents the parameters of the Beta distribution assumed by node *i* in its Bayesian view of node *j*'s behaviour. Initially, it is set to (1,1). So we assume *i* makes one individual observation about *j*; let s=1 if this observation is qualified as misbehaviour, and s=0 otherwise.

$$\alpha := \mu\alpha + s$$
$$\beta := \mu\beta + (1-s)$$

The weight μ serves as a fading mechanism, to discount for past experiences. This allows for node redemption. A node that behaved maliciously in the past and got bad reputation should be able to recover faster provided it presents a good cooperative behaviour. In order to estimate μ we use the following equation:

$$\mu := 1 - \frac{1}{m}$$

where *m* is the order of magnitude of the number of observations over which we believe it makes sense to assume stationary behaviour for the node. In addition, when the inactivity timer expires, we periodically reduce the values of α and β as follows:

$$\alpha := \mu\alpha$$
$$\beta := \mu\beta$$

This promotes sustained cooperative interactions, as otherwise the reputation of a node fades as time goes by without interactions. It also allows for node redemption even in the absence of interactions.

The reputation rating R_{ij} is also defined the same way as first-hand information record and initially it is also set to (1,1). It is updated on two types of events: when first-hand observation is updated and when a reputation rating published by some other node is received. Assume node *i* receives the reported first-hand information F_{kj} from node *k*. The question is how to detect and avoid false reports. For this we take into account trust and compatibility. First we need to take into account node *i*'s confidence on node *k*, which is measured by Trust rating T_{ik}

Trust rating T_{ik} uses a similar Bayesian approach, and has the form (γ,δ). It represents the parameters of the Beta distribution assumed by node *i* in its Bayesian view of node *k*'s trust. It is also, initially set to (γ,δ) = (1, 1). Then whenever a reputation rating update is received depending on the results of the deviation test defined below, the trust rating is modified. Let s=1 if the deviation test succeeds, and s=0 otherwise.

$$\gamma := \upsilon\gamma + s$$
$$\delta := \upsilon\delta + (1-s)$$

Here, υ is the fading factor of trust, similar to u. Also we perform the same inactivity updates as we do for first-hand information. Whenever the inactivity time expires, we reduce the values of γ and δ as follows:

$$\gamma := \upsilon\gamma$$
$$\delta := \upsilon\delta$$

In order to update the trust rating of the node, we perform the following deviation test:

$$\mathbb{E}(Beta(\alpha_F,\beta_F)) - \mathbb{E}(Beta(\alpha,\beta)) \geq d$$

Where:

- (α_F,β_F) is the value of F_{kj} as received in the reputation rating update from node *k*
- (α,β) is the value of R_{ij} locally calculated by node *i*
- d is a positive constant (deviation threshold)

The deviation test is positive, it is considered incorrect (as the expected reputation values are not compatible enough). Otherwise it is considered correct.

After performing the deviation test and updating node *k*'s trust rating T_{ik} the node is tested for trustworthiness, i.e. if it is considered confident, as follows:

$$\begin{cases} confident\ if\ \mathbb{E}(Beta(\gamma,\delta)) < t \\ not\ confident\ if\ \mathbb{E}(Beta(\gamma,\delta)) \geq t \end{cases}$$

The threshold *t* is an expression of tolerance depending on the scenario. If node *i* trusts a node *k* if it ratings deviate in no more than 15% of the cases, it should set *t* to 0.85. If either node *k* is considered confident or the deviation test is correct, the fist-hand information F_{kj} is considered compatible and the new reputation rating R_{ij} is updated as follows:

$$R_{ij} = R_{ij} + \omega F_{kj}$$

where ω is a small positive constant [20]. Otherwise the first-hand information is considered incompatible and is discarded. Finally, the decision-making process works as follows. Node *i* classifies the behaviour of node *j* as:

$$\begin{cases} regular\ \ if\ \mathbb{E}(Beta(\alpha,\beta)) < r \\ malicious\ \ if\ \mathbb{E}(Beta(\alpha,\beta)) \geq r \end{cases}$$

Again the threshold *r* is an expression of tolerance depending on the scenario.

```
Algorithm 8 Updating first-hand information table
if (non-cooperative interaction) then
   (s variable set to 1)
else
   (s variable set to 0)
end if
($\alpha^{NEW}$ = μ * α + s)
($\beta^{NEW}$ = μ * β + (1-s))
(save $\alpha^{NEW}$ and $\beta^{NEW}$ in the first-hand information table)
```

Algorithm 8: Updating first-hand information table

```
Algorithm 9 Obtaining α and β for the reputation table ϒ and δ
            for the confidence table
if (first-hand information received) then
   (update first-hand information table using previous algorithm)
   (update reputation table)
else
   for (each node ID contained in the second hand information received)
      (check deviation test for sender node information about node ID)
      (update confidence table for the sender node based on deviation test)
      (check the confidence of the sender node)
      if (sender node is confident) or (correct deviation test) then
         (update reputation table for node ED)
         (check new reputation for node ID)
         if (node ID not trustworthy) then
            (make alert)
            (send first-hand information table to all neighbour nodes)
         end if
      end if
   end for
end if
```

Algorithm 9: Obtaining α and β for the reputation table and γ and δ for confidence table

4.3.3.1 Demonstration Scenarios

The scenarios examined for demonstration purposes include devices from two node categories defined in the context of the nSHIELD project, i.e. micro, and Nano node. The aim is to demonstrate interoperability and the proposed schemes can be deployed even in constrained environments. In terms of applicability to the nSHIELD common demonstrators, Intrusion Detection in Wireless Sensor Networks finds its place within the Railways Security Scenario (T7.1).

5 Trusted and dependable Connectivity

The 802.15.4 specification is meant to support a variety of applications, many of which (such as nSHIELD), are security sensitive. For instance, in some sensor network' scenarios, there is a privacy concern about tracking the people in some building or facility. Additionally, if network is not secured and malicious user could inject and/or modify messages to cause a system failure. Many applications require confidentiality and most have a need for integrity protection. The protocol responsible for addressing these needs through a link-layer security is the 802.15.4.

Trusted and dependable connectivity are two important features of a secure network. In order to accomplish these goals, main activities performed in this field have been based on the study of the security (and general) requirements for lightweight link-layer communications in wireless sensor networks (WSN) scenarios.

Trusted and dependable, are two crucial features in secure networks, but not the only. Confidentiality, integrity and authenticity are also very important in order to communicate the nodes in a secure way, where the transmitted data is not compromised.

Providing a secure WSN is too generic. In order to accomplish this challenge, security has to be faced not only at the network layer but also in other layers of the OSI (or TCP/IP) reference model, represented in the table below.

Table 5-1: OSI and TCP/IP reference models

Application	Application
Presentation	
Session	
Transport	Transport
Network	Internet (Network)
Data Link	Link
Physical	

Security protocols exist at essentially OSI stack layer, plus some “in between” layers. Selecting the appropriate for the threats to be addressed requires attention to detail.

In the next subsections, a discussion of protocols and algorithms proposed to provide security in different stack layers are introduced. Moreover at the end of each section it has been proposed a demonstrator in order to test the feasibility and to show the performance of the proposed solutions, again, in terms of security and power consumption.

5.1 Secure communication protocols on the link layer

5.1.1 802.15.4 Overview

A link layer security protocol provides four basic security services: access control, message integrity, message confidentiality and replay protection.

Access control means the link layer protocol should prevent unauthorized parties from participating in the network. A secure network should provide message integrity protection: if an adversary modifies a message from an authorized sender while the message is in transit, the receiver should be able to detect this tampering. Including a message authentication code (MAC) with each packet provides message authentication and integrity.

Confidentiality means keeping information secret from unauthorized parties. It is typically achieved with encryption. Preferably, an encryption scheme should not only prevent message recovery, but also prevent adversaries from learning even partial information about the messages that have been encrypted. This stronger property is known as semantic security.

An adversary that eavesdrops on a legitimate message sent between two authorized nodes and replays it at some later time engages in a replay attack. Since the message originated from an authorized sender it will have a valid MAC, so the receiver will accept it again. Replay protection prevents these types of attacks. The sender typically assigns a monotonically increasing sequence number to each packet and the receiver rejects packets with smaller sequence numbers than it has already seen.

The 802.15.4 security layer is handled at the media access control layer, below application control. In order to specify the security requirements, appropriate control parameters have to be set into the radio stack. If the application avoids parameters, then the security is not activated. Then, applications must define security through control parameters. An application has a choice of security suites that controls the type of security protection that is provided for the transmitted data. Each security suite offers a different set of security properties. The 802.15.4 specification defines eight different security suites showed in next table (extracted from 802.15.4 specification). Note that the specification must support AES-CCM-64 and Null suites, being the rest optional.

Table 5-2: 802.15.4 security suites

Name	Description of the security level
Null	Avoiding security
AES-CTR	Just encryption
AES-CBC-MAC-128	MAC size of 16 Bytes (128 bits)
AES-CBC-MAC-64	MAC size of 8 Bytes (64 bits)
AES-CBC-MAC-32	MAC size of 4 Bytes (32 bits)
AES-CCM-128	Enable encryption and MAC size of 16 Bytes (128 bits)
AES-CCM-64	Enable encryption and MAC size of 8 Bytes (64 bits)
AES-CCM-32	Enable encryption and MAC size of 4 Bytes (32 bits)

Symmetric cryptography relies on both endpoints using the same key when communicating securely. In a group of nodes, the keying model governs what key a node uses to communicate with another node. The keying model that is most appropriate for an application depends on the threat model that an application faces and what types of resources it is willing to expend for key management. For example, in the network shared key model, every node uses the same key for communicating with every other node. Each node only needs to keep track of a single key, which eases the management problems. For pairwise keying, limits the scope of every key, where each pair of nodes shares a different key. Group keying is a compromise between network-shared and pairwise keying; a single key is shared among a set of nodes and used on all links between any two nodes in that group. The partition into groups may be made based on location, network topology, or similarity of function. Finally, hybrid approaches may use a combination of the above keying models simultaneously in the same application; for instance, pairwise keying to communicate to a base station and a network shared key for all the links in the WSN.

In the following table, we highlight the advantages and disadvantages of different keying models summarized above.

Table 5-3: Keying models comparison

Keying Model	Advantages	Disadvantages
Network shared keying	• Trivial key management. • Minimal memory requirements. • Applications can use the network shared key with little effort.	• Vulnerability to insider attacks. • A single internal node can break the confidentiality of any message.

Pairwise keying	• A node compromise only affects past and future messages sent to or from that node; other traffic is unaffected. • Provide better security than network shared keying.	• Overhead in key management. If a node communicates with many other nodes, it must store many keys and select the appropriate one when communicating. • Devices with minimal resources, the storage costs can be prohibitive.
Group keying	• Provides an intermediate tradeoff between networks shared keying and pairwise keying.	• Lower management and memory costs than pairwise keying but higher than network shared keying.

5.1.2 802.15.4 Nodes overview

Before defining secure communications protocols on the network and link layer, we have to define what constrained and unconstrained nodes are, for the Trusted and dependable Connectivity.

We define **Constrained nodes** as those with enough power and computational capabilities to execute the reference constrained OS such as Contiki OS or Tiny OS. In WP3 "SPD Node" these nodes are named as Micro or Nano nodes.

More specifically, Ed. C. Bormann gathered in the Internet-Draft "Guidance for Light-Weight Implementations of the Internet Protocol Suite draft-ietf-iwig-guidance-02" two class designations for light-weight nodes, according to this text, class designations may be used as rough indications of device capabilities:

Table 5-4: Class designations for light-weight nodes

Name	Data size (e.g., RAM)	Code size (e.g., Flash)
Class 1	~ 10 KiB	~ 100 KiB
Class 2	~ 50 KiB	~ 250 KiB

As we have introduced in previous deliverables/sections/work packages (SPD node), Zolertia Z1 (see Figure 5-1) is one of the selected reference hardware platform selected for nSHIELD project. The characteristics of this sensor are:

Zolertia Z1
Data size (RAM) = 8KiB
Code size (Flash) = 92KiB

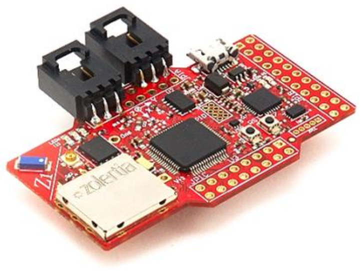


Figure 5-1: Zolertia Z1 sensor

The operating systems supported by Zolertia Z1 are Contiki OS and Tiny OS. We are going to highlight the most important features of both.

Contiki OS is an open source operating system for the Internet of Things. Contiki allows tiny, battery-operated low-power systems communicate with the Internet. Contiki is used in a wide variety of systems such as city sound monitoring, street lights, networked electrical power meters, industrial monitoring, radiation monitoring, construction site monitoring, alarm systems, and remote house monitoring. Contiki provides powerful low-power Internet communication.

Moreover, Contiki supports fully standard IPv6 and IPv4, along with the recent low-power wireless standards: 6lowpan, RPL, CoAP. With Contiki's ContikiMAC and sleepy routers, even wireless routers can be battery-operated. With Contiki, application development is written in standard C, with the Cooja simulator Contiki networks can be emulated before burned into hardware, and Instant Contiki provides an entire development environment in a single download. Contiki runs on a range of low-power wireless devices, many of which can be easily purchased online.

Contiki is developed by a developers with contributions from Atmel, Cisco, ETH, Redwire LLC, SAP, SICS, Thingsquare, and many others, led by Adam Dunkels of Thingsquare. Contiki is open source software: Contiki can be freely used both in commercial and non-commercial systems and the full source code is available.

TinyOS is an open source, BSD-licensed operating system designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing, personal area networks, smart buildings, and smart meters. A worldwide community from academia and industry use, develop, and support the operating system as well as its associated tools, averaging 35,000 downloads a year.

TinyOS applications are written in nesC, a dialect of the C language optimized for the memory limits of sensor networks. Its supplementary tools are mainly in the form of Java and shell script front-ends. Associated libraries and tools, such as the NesC compiler and Atmel AVR bin utils toolchains, are mostly written in C.

TinyOS programs are built out of software components, some of which present hardware abstractions. Components are connected to each other using interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage.

TinyOS is completely non-blocking: it has one stack. Therefore, all I/O operations that last longer than a few hundred microseconds are asynchronous and have a callback. To enable the native compiler to better optimize across call boundaries, TinyOS uses nesC's features to link these callbacks, called events, statically. While being non-blocking enables TinyOS to maintain high concurrency with one stack, it forces programmers to write complex logic by stitching together many small event handlers. To support larger computations, TinyOS provides tasks, which are similar to a Deferred Procedure Call and interrupt [21] handler bottom halves. A TinyOS component can post a task, which the OS will schedule to run later. Tasks are non-preemptive and run in FIFO order. This simple concurrency model is typically sufficient for I/O centric applications, but its difficulty with CPU-heavy applications has led to the development of a thread library for the OS, named TOS Threads.

TinyOS code is statically linked with program code, and compiled into a small binary, using a custom GNU toolchain. Associated utilities are provided to complete a development platform for working with TinyOS.

We define **Unconstrained nodes**, as those with enough power and computational capabilities to execute a Linux-based distribution.

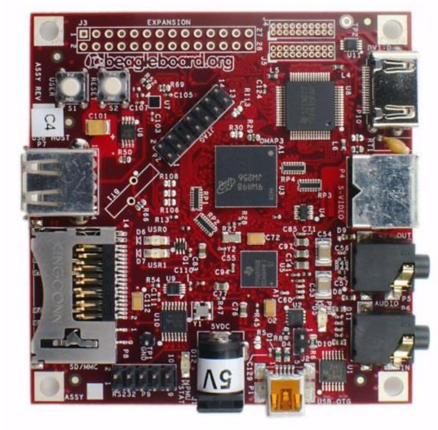
Today, Linux systems are used in every domain, from embedded systems to supercomputers, and have secured a place in server installations. Use of Linux distributions in home and enterprise desktops has been growing. They have also gained popularity with various local and national governments. Linux distributions have also become popular in the netbook market, with many devices such shipping with customized Linux distributions installed.

With the availability of consumer embedded devices, communities of users and developers were formed around these devices: Replacement or enhancements of the Linux distribution shipped on the device has often been made possible thanks to availability of the source code and to the communities surrounding the devices.

The advantages of embedded Linux over proprietary embedded operating systems include multiple suppliers for software, development and support; no royalties or licensing fees; a stable kernel; and the ability to read, modify and redistribute the source code.

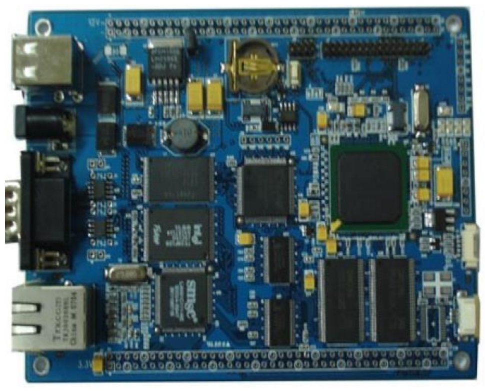
Examples of these types of nodes are:

Beagle Bone



Beagle Board MX

RaspBerry Pi



OMBRA-nSHIELD Board (SES)

Figure 5-2: Unconstrained nodes

Taking into account the previous considerations, we will try to cover security in the data link and network layers for the previously described devices.

5.1.3 Link layer security

Network Access Control (NAC) refers to methods used to authorize or deny network communications to particular systems or users. In other words, our system before transmitting data through the network has to ensure that new nodes can join to and leave from the network in a secure way.

One of the protocols to manage NAC is EAP. EAP can be used with multiple link-layer technologies and supports multiple methods for implementing authentication, authorization, and accounting (AAA). EAP does not perform encryption itself, so it must be used in conjunction with some other cryptographically strong protocol to be secure. EAP uses the same concepts of supplicant and authentication server as does 802.1X[1], but with different terminology.

One of the advantages of the EAP architecture, represented in the following figure, is its flexibility. EAP is used to select a specific authentication mechanism, typically after the authenticator requests more information in order to determine the specific authentication method to be used. Rather than requiring the authenticator to be updated to support each new authentication method, EAP permits the use of a

[1] **IEEE 802.1X** is an IEEE Standard for port-based Network Access Control (PNAC). It is part of the IEEE 802.1 group of networking protocols. It provides an authentication mechanism to devices wishing to attach to a LAN or WLAN.

backend authentication server, which may implement some or all authentication methods, with the authenticator acting as a pass-through for some or all methods and peers.

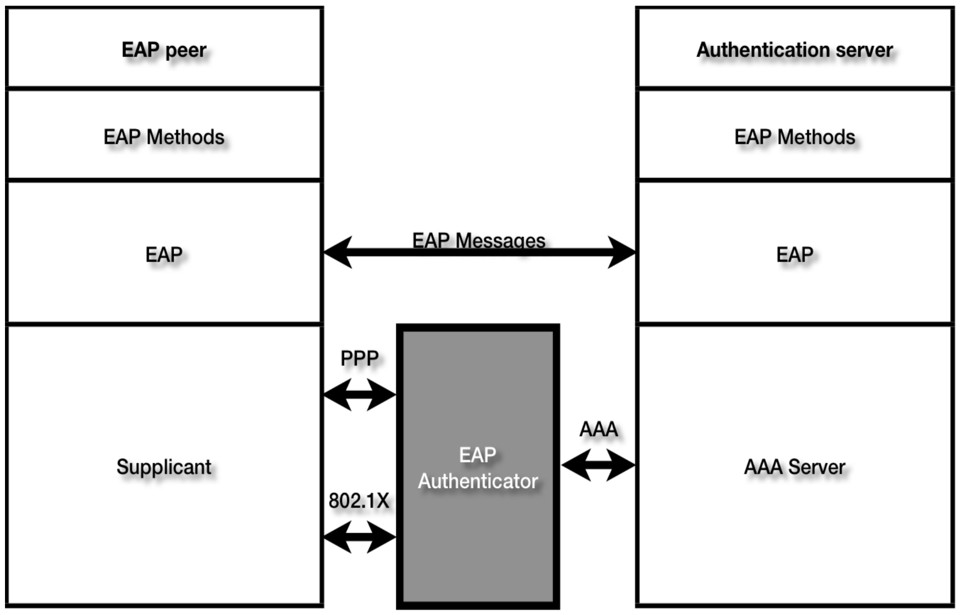


Figure 5-3: EAP Architecture

The terminology of the EAP Architecture is defined below:

- **EAP peer:** The end of the link that responds to the authenticator. In [IEEE-802.1X], this end is known as the Supplicant.
- **EAP Methods:** EAP provides some common functions and negotiation of authentication methods called EAP methods. There are currently about 40 methods defined. Methods defined in IETF RFCs include: EAP-MD5, EAP-POTP, EAP-GTC, EAP-TLS, EAP-IKEv2, EAP-SIM, EAP-AKA, and in addition a number of vendor specific methods and new proposals exist. Commonly used modern methods capable of operating in wireless networks include: EAP-TLS, EAP-SIM, EAP-AKA, LEAP and EAP-TTLS. Requirements for EAP methods used in WLAN authentication are described in RFC 4017.
- **Supplicant:** The end of the link that responds to the authenticator in [IEEE-802.1X].
- **EAP Authenticator:** The end of the link initiating the EAP authentication. This term is used in [IEEE-802.1X].
- **Authenticator server:** the entity that terminates the EAP authentication method with the peer. In the case where no backend authentication server is used, the EAP server is part of the authenticator. In case where the authenticator operates in pass-through mode, the EAP server is located on the backend authentication server.
- **AAA Server:** Authentication, Authorization, and Accounting. AAA protocols with EAP support include RADIUS [RFC3579] and Diameter [DIAM-EAP]. The term “AAA Server” is equivalent to “backend server”.

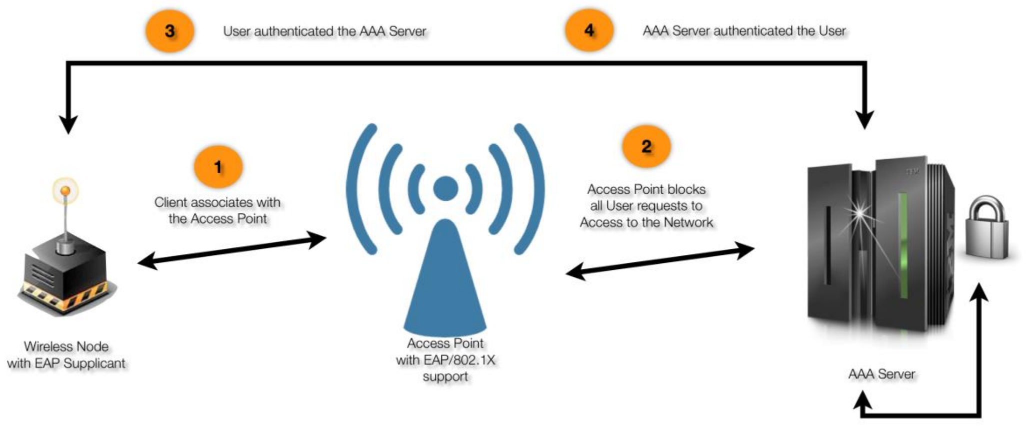

3
User authenticated the AAA Server
4
AAA Server authenticated the User
1
Client associates with the Access Point
2
Access Point blocks all User requests to Access to the Network
Wireless Node with EAP Supplicant
Access Point with EAP/802.1X support
AAA Server

Figure 5-4: EAP simplified authentication process (Wireless)

Given the EAP terminology, in the above Figure is pointed out the simplified scenario of the EAP authentication process. These are:

1. Client associates with the Access point. In order to request access to the network (wireless in this case).
2. Access point requests identification data from the Client.
3. Access Point blocks all User requests to Access to the network.
4. The AAA Server is authenticated by the Client according to its credentials
5. Finally the Client is identified by the AAA Server according to its credentials.

In order to ensure the SPD criteria for both the Client and the AAA Server, we propose a link layer access mechanism based on the certificates exchange between nodes. In order to accomplish this task, the certificate exchange should be carried by the extensible authentication protocol when the method applied is TLS or (PEAP with EAP-TLS).

Before going deeper on the details of this proposal, we will explain the certificate management through the following simplified scenario.

In the case of applying strong EAP methods such as TLS, with certificates, both the client (the node requesting access to the network) and the server (the AAA server) use certificates to verify their identities to each other.

Certificates must meet specific requirements both on the server and on the client for successful authentication. All certificates that are used for network access authentication must meet the requirements for X.509 certificates, and they must also meet the requirements for connections that use Secure Sockets Layer (SSL) encryption and Transport Level Security (TLS) encryption.

In next subsection we are going to explain more in details the certificate management system given that the certificates will be used to ensure confiability and security in the link and network layer.

5.1.3.1 Certificate Management System

To ensure security and confiability in the WSN, i.e. to certify the identity of all nodes of the network and to secure the data transmission among the nodes, certificates must be part of the security framework provided by nSHIELD.

One of the nSHIELD's efforts is to base the proposed solutions on standards (or tacitly standardized solutions) in order to manage the certificates in the project we are going to use the OpenSSL library.

OpenSSL library is common used among the research and commercial community. The project is managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation. This full-featured and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library.

5.1.3.2 Proposed solution

To be as near as possible to the 802.15.4 specifications and avoid market fragmentation, the proposed solution is to implement Counter Mode Encryption (CTR), Cipher Block Chaining Message Authentication Code (CBC-MAC) and counter with CBC-MAC (CCM) algorithms over the over the 128-bit AES encryption provided by modern transceivers.

The device we are going to use is the Zolertia z1, as described on the “nodes overview” section. And the operative system selected to enhance the security cc2420 layer will be TinyOS.

CTR

This algorithm provides confidentiality using the block cipher algorithm AES. The sender breaks the plaintext into blocks of 16 bytes and computes $c_i = p_i \oplus Ek\ (x_i)$, being p_i a block of plaintext, c_i the cipher text, and x_i the value of a counter used for the block number i.

This is to say, a block cipher is used to produce a flow known as pseudorandom key stream. This stream is combined with the plaintext by XOR leading encryption.

The fields added using this method are a 4 bytes frame counter (FC) and 1-byte key counter (KC) which is incremented when the frame counter arrives to the maximum.

So the frame we obtain after this method is like:

FC (4 bytes)	KC (1 byte)	Ciphered data (variable)

CBC-MAC

This mode provides integrity and authentication using CBC-MAC algorithm. The transceiver can generate a 4, 8 or 16 bytes MAC code. The MAC can be calculated by entities which share the same symmetric key. This code protects both data and headers of the frame. The sender adds the MAC to the plaintext. The receiver verifies the MAC code by calculating the code from the received frame and comparing the value contained in that frame. The verification of this code allows testing if data was not altered, if the message comes from the correct sender and if the message is in the correct order.

The frame of this type of algorithm is like this:

Data (variable)	MAC (4,8 or 16 bytes)

CCM

This mode is a combination of CTR and CBC-MAC. First, CBC-MAC algorithm is used to provide integrity protection and after that, data and MAC are ciphered with the CTR algorithm. Going to the frame, we find both, frame and key counters and MAC.

FC	KC	Ciphered data (variable)	Ciphered MAC (4,8,16)

Analysing the quantity of useful data we can obtain applying these algorithms we can see:

Table 5-5: Frames length using different security levels keying models comparison

n	NO SEC	CTR	CBC-MAC-4	CBC-MAC-8	CBC-MAC-16	CCM4	CCM8	CCM16
60	74	79	78	82	90	83	87	95
70	84	89	88	92	100	93	97	105
80	94	99	98	102	110	103	107	115
90	104	109	108	112	120	113	117	125
95	109	114	113	117	125	118	122	130
100	114	119	118	122	130	123	127	135
110	124	129	128	132	140	133	137	145
115	129	134	133	137	145	138	142	150

The maximum length a frame could have is 127 bytes, headers included. Taking this into account we can see that if we don't use security the maximum data length is 113 bytes while if we apply the stricter security (CCM16) only 92 bytes of data could be sent.

Going further, is easy to guess that the more security we apply the more energy we need to send data and the more time to encrypt and decrypt the information as we can check on Carolina Trip investigation about the number of frames sent with a battery applying different security levels

Table 5-6: Number of frames sent with an AA battery using different security algorithms

	Number of frames ($x10^6$)	
bytes	**NOSEC**	**CCM16**
60	116	98
65	109	93
70	103	89
75	97	85
80	92	81
85	87	78
90	83	75

Taking into account the outlined numbers, the costs are very modest compared to the security provided. Then, these algorithms are suitable for WSN scenarios where performance and security must be guaranteed.

5.1.4 Demonstration scenarios

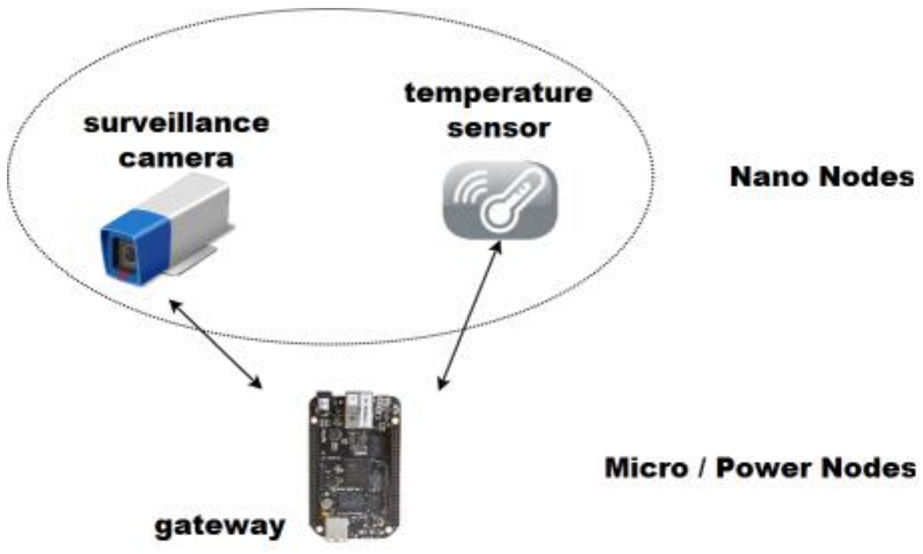


Figure 5-5: Link layer security demonstration scenarios

We want to demonstrate the performance of a particular WSN composed of:

- Two Nano nodes: sensors with TinyOs measuring temperature, humidity or other environment parameter.
- One micro or power node: the gateway of the WSN to the rest of networks (i.e. Internet).

In order to test the difference security and power consumption level, we are going to plan the following configurations:

1. TinyOs without TinySec.
2. TinySec enabling Authentication Encryption.
3. TinySec enabling Authentication Only.

Note that, the WSN can be composed by several sensors, each running TinyOS and measuring any relevant environment parameter for a specific environment/scenario.

5.2 Secure communication protocols on the network layer

5.2.1 Description

When dealing with communications among nSHIELD nodes, security on exchanged messages can be applied to most of the layers of the TCP/IP stack, with the most prominent being the application, network and data link layer. The corresponding security mechanisms for these three layers are the well-known Transport Layer Security Protocol (TLS) and its compact variant, namely Datagram Transport Layer Security Protocol (DTLS), Internet Protocol Security (IPsec) for the network layer and its encoded version for 6lowPAN and, if IEEE802.15.4 is the underlying data link layer protocol, the inherent security mechanism defined in the standard.

Each of these mechanisms has benefits and drawbacks and satisfies different needs. Providing security at lower layers of the network stack relieves applications from deploying their own distinct security mechanisms to protect communications. However, this comes at a cost. Security at the data link layer introduces significant computational overhead to routing nodes as messages are protected at the lowest layer, and therefore, protection takes place on a node-by-node basis. Therefore, messages have to be decrypted, verified and re-encrypted, typically using a different set of keys, prior to being forwarded to the next node. This process consumes valuable node resources for routing nodes.

IPsec, as opposed to 802.15.4, offers end to end protection. Therefore, routing nodes resources are only used for message routing. Moreover, IPsec can offer confidentiality, integrity and message authentication to all application data. For these reasons, security at the network layer can be considered as a valuable resource for low power and lossy networks. Note that IPsec when applied to a WSN cannot provide traffic flow confidentiality as defined in IPsec ESP, since it requires the use of tunnel mode to conceal the addresses of source and destination and additional space for padding to conceal communication patterns and characteristics.

Another advantage of using IPsec is that it offers communicating parties the ability to choose among authentication, encryption, or authentication and encryption, based on the corresponding communications needs. More specifically, AH can only provide authentication and integrity to exchanged messages, while ESP can provide confidentiality and, optionally, authentication to a reduced set of header fields compared to AH. One of the proposed options in terms of algorithms is to use ESP with AES in Counter with CBC-MAC (CCM) mode [22] as the algorithm to provide encryption and authentication. It is worth pointing out that the functionality of offering only authentication is not supported by AES-CCM, according to the original description of the cipher. This feature was proposed in the version named CCM* (CCM-Star), which is defined in [23] and in effect separates the mechanism of message encryption from that of integrity checking and allows them to be called independently. In this way, a greater variety of security levels can be offered, as shown in Table 5-7.

Table 5-7: Offered security levels

Security Level	Security Attributes	Data Confidentiality	Data Authenticity
0	None	OFF	NO
1	MIC-32	OFF	YES
2	MIC-64	OFF	YES
3	MIC-128	OFF	YES
4	ENC	ON	NO
5	ENC-MIC-32	ON	YES
6	ENC-MIC-64	ON	YES
7	ENC-MIC-128	ON	YES

Given the constrained environment that these communications take place, deploying the IPsec ESP and AH protocols designed for powerful devices is not an acceptable solution, mainly due to the extended length of header data required to convey protocol information. The restrictions imposed by the IEEE802.15.4 message length drive the introduction of compressed header solutions with encoded fields. Such a solution has been proposed in [24] and [25]. This is the first of the two schemes examined for the needs of this deliverable and defines a compressed format of IPsec ESP and AH protocols. A variant of the first solution which takes advantage of the benefits provided by combining (compressed) IPsec ESP protocol with AES in CCM* mode comprises the second scheme examined here.

The aim of the work undertaken in this field is two-fold:

1. Provide an efficient solution for securing communications. Emphasis is given on highly constrained nodes and their capacity to accommodate the proposed scheme.
2. Use benchmarks to compare the proposed solution with other schemes that operate on the same or other layers.

Prior to giving the details of the two schemes it is important to provide some of the benefits of using AES in CCM mode to justify the proposal made for an alternative to the first scheme:

- AES_CCM* is the algorithm used by IEEE 802.15.4 to provide security at the data link layer, and therefore, LoWPAN Nodes are bound to support it and might also bear hardware implementations for AES_CCM* encryption, decryption and integrity check processing. Assuming the availability of

IEEE802.15.4, regardless the implementation choice of this interface, i.e. hardware or software, the crypto functionality is expected to be available to upper layers' software to use it.

- In contrast to other block cipher modes which typically offer only encryption, CCM is an authenticate-and-encrypt block cipher mode [26] and can be used with ESP, to provide confidentiality, data origin authentication and integrity. The underlying block cipher must have 128-bit blocks and therefore AES is a strong candidate to use in CCM* mode.
- It requires minimal message expansion (caused by the Initialization Vector and the Integrity Check Value), it only requires a single key and it can handle messages of arbitrary length, i.e. it does not require padding.

5.2.2 Demonstration Scenarios

The scenarios examined for demonstration purposes include devices from all three node categories defined in the context of the nSHIELD project, i.e. power, micro, and Nano node. The aim is to demonstrate interoperability and that such the proposed schemes can be deployed even in highly constrained environments.

There are different scenarios with different characteristics regarding the use of IPsec by nSHIELD nodes that can be examined. Examples include the following and are based on the use of Wireless Sensor Network where the distinct nodes are nSHIELD nodes.

1. Node-to-node within the same WSN. The proposed schemes allow any two nodes, such as the sink node or any other node, to communicate securely using IPsec, hence hiding the communicated data from intermediate routing nodes.
2. Node to remote party without gateway. The remote party has to support the compressed IPsec ESP format using AES_CCM* as proposed in this document.
3. Node to remote party through gateway. The remote party has to support the IPsec ESP mode using AES_CCM* as defined in [23], while the gateway, supporting both AES_CCM* and the scheme proposed in section 5.2.1, needs to transform these messages accordingly prior to forwarding them to the other end.

Note that the above scenarios will not demonstrate interoperability among the two schemes described in section 5.2.1. The two implementations are accomplished in parallel mainly to use performance results for comparison.

5.3 Access control in Smart Grid networks

nSHIELD will deal with different devices and embedded systems interacting though low and medium power lines.

Different components encompass the Smart grid: Smart Meters, home gateway, system operation management elements, disturbances and fluctuation sensors, processors, etc. As we are evolving towards an Information Utility, it is necessary to protect all information processing and travelling across the Smart grid.

Nowadays Smart grids are considered a fundamental Critical Infrastructure which enables energy distribution across cities. Energy supply is considered a basic necessity for citizen and industry and its denial can incur in fatal accidents and moreover mortalities.

This use case is about enabling PKI infrastructure through Smart grid miscellaneous sub-systems, components and embedded systems and implementing and certifying a reliable and dependable Smart grid.

5.3.1 Essential technologies

This use case will make emphasis on the following technologies:

- Credential management full life-cycle methodology creation for Smart grid

- Focus on Secure Elements (SE in FIPS140-anti tampering properties) for digital certificates storage. SEs should be SIM cards or Cryptographic SD cards. In case there is not SEs, we need to have robust and secured SW containers for credential management.
- We are addressing privacy related issues in low voltage lines: Anonymisation and citizen empowerment extended privacy
- Security Certification for DLMS protocols

5.3.2 Security, privacy and dependability demands

Smart meters manufacturers and DSOs often introduce security on a second phase after operational declarations. Control access based mechanisms have only been addressed (being them quite weak) Future information operators need to encrypt information for having a correct measurement fulfilling integrity and confidentiality requirements satisfied.

There are two basic requirements, depending on the type of power line:

- Low power line: [REQ 01] Privacy maintaining. All users and citizen will preserve privacy
- Medium power line: [REQ 02] No sabotages through encryption for RTUs and concentrators.

For these requirements it is necessary to develop a PKI based architecture which provides M2M IAA (identity, Authentication and Authorisations), guaranteeing integrity and confidentiality. This should be a plug &Play appliance connected to the Smart Grid system operation management that provides full certificates lifecycle support.

Figure 5-6: TECNALIA Laboratories (INGRID)

There is also an INDUSTRIAL need for certifying security in low power line: TECNALIA will develop tools for certifying security, privacy and dependability (some attributes) for protocols such as DLMS.

Therefore nSHIELD will specify a SECURITY Certification criterion for the information controlled by DLSM/COSEM protocol. This will assure:

- Correct test for:
  - Association for control access
  - Penetration tests
- Mechanism based on PKI infrastructure

<u>DLMS Cosem</u>

“Automatic Meter Reading, or more general - Demand Side Management - needs universal definitions and communication standards. DLMS/COSEM is the common language so that the partners can understand each other.”

nSHIELD will develop tests and tools to measure security weaknesses and develop security mechanisms in order to have a secured connectivity in the Automatic Meter reading and connection to concentrators in higher level of the net as a Network prototype.

Association algorithms

The DLMS application protocol is a connection-oriented (CO) protocol; it means that before sending data between client and server they need to “associate”.

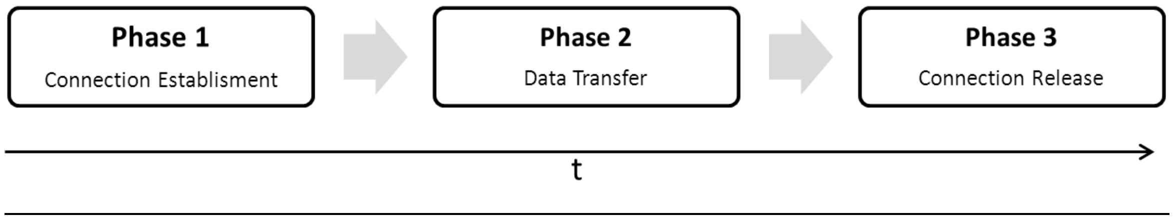


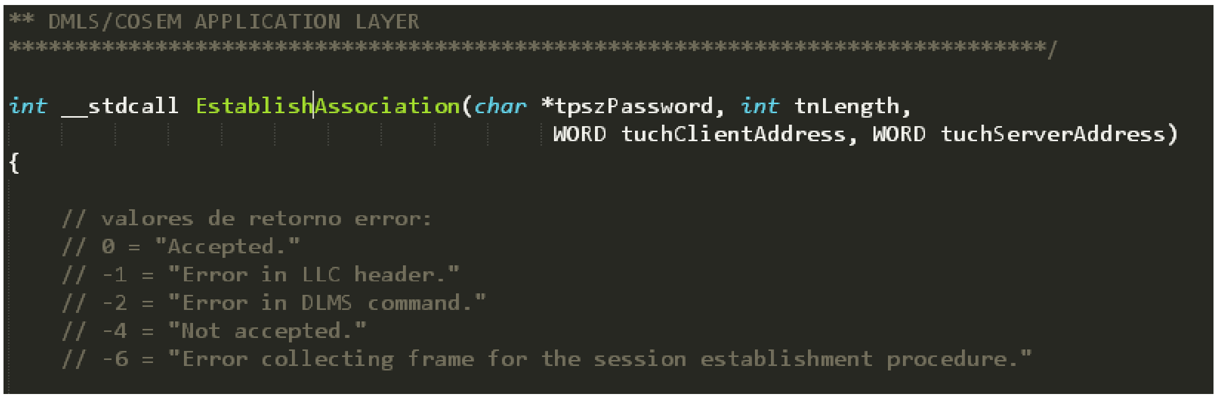
Figure 5-7: DLMS connection phases

DLMS has three levels of security:

- Lowest level security (no security): This level allows the client to retrieve basic information.
- Low Level Security (LLS): LLS allows the client to retrieve information supplying the correct password in the connection establishment (Phase 1).
- High Level Security (HLS): HLS allows mutual authentication of the client and the server participating in an association.

Nowadays security levels mostly used are “no security” and LLS levels. TECNALIA has developed several C++ libraries of DLMS protocol specification.

Association establishment function using LLS security level is depicted as follows:



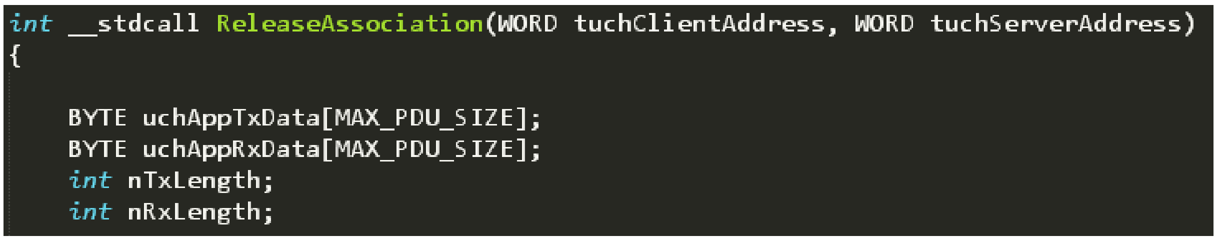
```
** DMLS/COSEM APPLICATION LAYER
**************************************************************************/

int __stdcall EstablishAssociation(char *tpszPassword, int tnLength,
                                   WORD tuchClientAddress, WORD tuchServerAddress)
{

    // valores de retorno error:
    // 0 = "Accepted."
    // -1 = "Error in LLC header."
    // -2 = "Error in DLMS command."
    // -4 = "Not accepted."
    // -6 = "Error collecting frame for the session establishment procedure."
```

Figure 5-8: C++ Association Establishment Code

Release association function is in the following figure:



```
int __stdcall ReleaseAssociation(WORD tuchClientAddress, WORD tuchServerAddress)
{

    BYTE uchAppTxData[MAX_PDU_SIZE];
    BYTE uchAppRxData[MAX_PDU_SIZE];
    int nTxLength;
    int nRxLength;
```

Figure 5-9: C++ Release Association Code

The main important issue is to extend these association techniques to those classes that implement security setup call for encryption and converting LLS in HLS (High Level Security). nSHIELD project may help to improve this implementation and provide to PRIME/DLMS community (standards community) an improvement from nSHIELD layered perspective.

Control access in DLMS has 4 authorisation profiles, whereas authentication is basic (BASIC_AUTH). Increasing security could decrease performance status. Therefore there is a trade-off between security and performance that needs to be analysed.

References

[1] P. Morerio, K. Dabcevic, L. Marcenaro and C. Regazzoni, “Distributed cognitive radio architecture with automatic frequency switching,” in *Complexity in Engineering (COMPENG) pp. 1-4*, 2012.

[2] K. Dabcevic, L. Marcenaro and C. Regazzoni, “Reputation-based frequency switching algorithm for defense against intelligent jamming attacks in centralized Cognitive Radio Networks,” in *submitted for IEEE International Conference on Sensing, Communication, and Networking (SECON 2013)*, 2013.

[3] S. K. Singh, M. P. Singh y D. K. Singh, «Routing Protocols in Wireless Sensor Netwroks – A Survey,» *In International Journal of Computer Science & Engineering Survey (IJCSES),* vol. 1, nº 2, pp. 63-83, November 2010.

[4] G. Hatzivasilis and C. Manifavas, “Building Trust in Ad hoc Distributed Resource-sharing Networks Using Reputation-based Systems,” in *In 16th Panhellenic Conference on Onformatics PCI (pp. 416-421)*, http://doi.ieeecomputersociety.org/10.1109/PCi.2012.28, 2012.

[5] S. Marti, T. J. Giuli, K. Lai and M. Baker, “ Mitigating Routing Misbehavior in Mobile Ad Hoc Networks,” in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom’00), pp. 255-265*, August 2000.

[6] S. Buchegger and J. L. Boudec, “Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes - Fairness In Dynamic Ad-hoc NeTworks),” in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc’02), pp. 226-336*, June 2002.

[7] S. Ganeriwal, L. Balzano and M. Srivastava, “Reputation-based framework for high integrity sensor networks,” in *In Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SAN ’04), pp. 66-77.*, October 2004.

[8] P. Michiardi and R. Molva, “Core: A Collaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks,” in *Communication and Multimedia Security Conference (CMS’02)*, September 2002.

[9] A. K. Trivedi, R. Arora, R. Kapoor, S. Sanyal and S. Sanyal, “A Semi-distributed Reputation-based Intrusion Detection System for Mobile Adhoc Networks,” Networks.arXiv:1006.1956v2, June 2010.

[10] A. T. Rahem and H. K. Sawant, “Collaborative Trust-based Secure Routing based Ad-hoc Routing Protocol,” *International Journal of Modern Engineering Research (IJMER),* vol. 2, no. 2, pp. 095-101, Mar-Apr 2012.

[11] Y. Zhang, L. Xu and X. Wang, “A Cooperative Secure Routing Protocol based on Reputation System for Ad Hoc Networks,” *Journal of Communications,* vol. 3, no. 6, pp. 43-50, November 2008.

[12] S. Madhavi and T. H. Kim, “An Intelligent Distributed Reputation Based Mobile Intrusion Detection System,” *International Journal of Computer Science and Telecommunications,* vol. 2, no. 7, October 2011.

[13] A. Pirzada y C. McDonald, «Trust Establishment in Pure Ad-hoc Networks,» *Wireless Personal Communications,* vol. 37, pp. 139-163, 2006.

[14] B. Karp and H. T. Kung, “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks,” in *In Proceedings of the 6th ACM/IEEE Annual International Conference on Mobile Computing and Networking (MoniCom’00), pp. 243-254*, Aug. 2000.

[15] K. Gerrigagoitia, R. Uribeetxeberria, U. Zurutuza and I. Arenaza, “Reputation-based Intrusion Detection System for wireless sensor networks,” in *Complexity in Engineering (COMPENG)*, 2012.

[16] S. Buchegger and J. Y. L. Boudec, “A robust reputation system for mobile ad-hoc networks,” in *Proceedings of P2PEcon*, 2003.

[17] A. Jsang and R. Ismail, “The beta reputation system,” in *in Proceedings of the 15th Bled Electronic Commerce Conference (pp. 41–55)*, 2002.

[18] S. Ganeriwal, L. K. Balzano and M. B. Srivastava, “Reputation-based framework for high integrity sensor networks,” *ACM Transactions on Sensor Networks (TOSN),* vol. 4, no. 3, p. 15, 2008.

[19] S. Buchegge and J. L. Boudec, “A Robust Reputation System for Mobile Ad-hoc Networks,” EPFL IC Technical Report IC (p. 50), 2003.

[20] J. O. Berger, Statistical Decision Theory and Bayesian Analysis, Springer, second edition, 1985.

[21] Wikipedia, “Interrupt handler,” [Online]. Available: http://en.wikipedia.org/wiki/Interrupt_handler. [Accessed 2013].

[22] R. Housley, “Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP),” RFC 4309.

[23] R. Struik, “Formal specification of the CCM* mode of operation,” IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs), 2005.

[24] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt and U. Roedig., “Securing Communication in 6LoWPAN with Compressed IPsec,” in *7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '11)*, Barcelona, Spain, June 2011.

[25] S. Raza, T. Voigt and U. Roedig, “6LoWPAN Extension for IPsec.,” in *Interconnecting Smart Objects with the Internet Workshop*, Prague, Czech Republic, March 2011.

[26] D. Whiting, R. Housley and N. Ferguson, “Counter with CBC-MAC (CCM),” RFC 3610, September 2003.

[27] J. Hui and E. P. Thubert, “Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks,” RFC 6282.

[28] A. Srinivasan, J. Teitelbaum and J. Wu, “Drbts: Distributed reputation based beacon trust system,” in *In 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, pp. 277–283*, 2006.

[29] S. Buchegger and J. Y. L. Boudec, “A Robust Reputation System for Peer-to-Peer and Mobile Ad Hoc Networks,” in *In Proceedings of P2Pecon*, Harvard University, Cambridge MA, USA, June 2004.

[30] C. Tripp, “Impact of security mechanisms in IEEE 802.15.4 sensor operationv,” 2009.