



<FP7-ICT-STREP>

Contract No. 258280

TWISNet

Trustworthy Wireless Industrial Sensor Networks

Deliverable D3.4.2

Concepts and design of protocols for communication, information and service availability

Contractual date:	M30
Actual submission date:	M30
Responsible beneficiary:	HTW
Authors:	Alex Olteanu (UPB), Alexis Olivereau (CEA), Basil Hess (SAP), Dan Dragomir (UPB), Dr. Dan Tudose (UPB), Dr. Emil Slusanschi (UPB), Enrico Lehmann (DDE), Dr. Felix von Reischach (SAP), Laura Gheorghe (UPB), Markus Wehner (HTW), Mike Ludwig (DDE), Nouha Oualha (CEA), Oliver Kasten (SAP), Silvia Stegaru (UPB), Dr. Sven Zeisberg (HTW)
Work package:	WP3
Security:	Public
Nature:	Report
Document name:	TWISNet_WP3_D342.docx
Version:	9.0

REVISION HISTORY

Date (dd.mm.yyyy)	Version	Author	Comments
04.04.2011	0.1	Markus Wehner	created template for chapter 1
25.04.2011	1.0	Laura Gheorghe	added Sota part of HTW
05.05.2011	2.0	Emil Slusanschi	added Sota part of UPB
05.05.2011	3.0	Alexis Olivereau	added Sota part of CEA
10.05.2011	4.0	Laura Gheorghe	updated Sota part of HTW
11.05.2011	5.0	Emil Slusanschi	updated Sota part of UPB
11.05.2011	6.0	Basil Hess	added Sota part of SAP
10.06.2011	7.0	Enrico Lehmann	added Sota part of DDE
14.08.2011	8.0	Laura Gheorghe	updated Sota part of HTW
22.08.2011	9.0	Markus Wehner	added structure for chapter 2
05.09.2011	10.0	Markus Wehner	updated Sota parts
28.09.2011	11.0	Markus Wehner	added sections 2.1, 2.2, added paragraphs to 2.3
24.10.2011	13.0	Alexis Olivereau	added(short) sections on self-healing
27.10.2011	14.0	Silvia Stegaru	added a paragraph to section 2.3.3 Failure detection, modified Figure 1 of section 2.2
18.11.2011	15.0	Markus Wehner	add TOC for chapter 3
20.02.2012	17.0	Silvia Stegaru	filled in section 3.3 (the FABD module). Modified section 2.3.3.
23.02.2012	18.0	Markus Wehner	added descriptions to chapter 3
24.02.2012	19.0	Markus Wehner	detailed description of chapter 3 modules RB and SFM
28.02.2012	20.0	Markus Wehner	review of the document
09.03.2012	21.0	Basil Hess	integrate changes according to review
14.03.2012	22.0	Silvia Stegaru	integrate changes according to review
15.03.2012	23.0	Alexis Olivereau	update of FA and SH
27.03.2012	24.0	Markus Wehner	harmonization of document
29.03.2012	25.0	Markus Wehner	F2F discussion changes
02.04.2012	26.0	Markus Wehner	further minor changes
02.04.2012	27.0	Dan Tudose	added description to missing references

13.04.2012	28.0	Mike Ludwig	final review
14.04.2012	29.0	Markus Wehner	finalization
25.01.2013	1.0	Markus Wehner	ToC and skeleton for D3.4.2 based on relevant D3.4.1 material
20.02.2013	2.0	Markus Wehner	updated SFM
06.03.2013	3.0	Markus Wehner	second rework SFM, updated RB
25.03.2013	4.0	Silvia Stegaru	Updated the FABD sections.
25.03.2013	5.0	Markus Wehner	general update
07.04.2013	6.0	Mike Ludwig	review
09.04.2013	7.0	Markus Wehner	update based on review
12.04.2013	8.0	Alexis Olivereau	update of SH, FA
14.04.2013	9.0	Markus Wehner	finalization

Abstract:

Wireless Sensor Networks have drawn the attention of the research community for decades. They provide applications in monitoring and controlling of industrial processes, workers, environmental phenomena, and many more. However, because sensors typically communicate over the air, there is always a certain danger that information can be accessed and modified by unauthorized parties. TWISNet (Trustworthy Wireless Industrial Sensor Networks) aims at addressing security issues arising when applying wireless sensor networks to industrial environments.

This document presents the concepts and solutions developed for communication, information and service availability in TWISNet FP7 WSN security project. The following technical themes are considered: (1) trustworthy assessment of system availability (failure anticipation, prevention, detection) and (2) system resilience solution, possibly in a multi-administrative domain environment. Therefore, an architecture is presented which employs the following modules: Sensor Failure Management, Failure Anticipation, Failure and Abnormal Behavior Detection, Resilience Backup and Self Healing.

To identify such modules, the current state of the art is examined. All the relevant topics as mentioned in the section above are covered in the analysis. The actual literature is presented with respect to collaborative projects, existing standards and available commercial products. A summary concludes the various topics of the research.

Keyword list:

trustworthy assessment, availability, failure anticipation, failure prevention, failure detection, resilience

TABLE OF CONTENTS

- List of Abbreviations..... 7**
- 1. Introduction..... 9**
- 2. State of the art 10**
 - 2.1 Research literature 10**
 - 2.1.1 Trustworthy assessment 10
 - 2.1.2 System resilience solutions 11
 - 2.2 Collaborative projects 18**
 - 2.2.1 Trustworthy assessment 18
 - 2.2.2 System resilience solutions 19
 - 2.3 Standards 20**
 - 2.3.1 Trustworthy assessment 20
 - 2.3.2 System resilience solutions 21
 - 2.4 Commercial products 21**
 - 2.5 Summary 22**
- 3. Architecture Functional Description 24**
 - 3.1 Overview 24**
 - 3.2 Presentation 24**
 - 3.3 Modules Functional Description 25**
 - 3.3.1 Sensor Failure Management 25
 - 3.3.2 Failure Anticipation..... 25
 - 3.3.3 Failure and Abnormal Behaviour Detection 25
 - 3.3.4 Resilience Backup..... 26
 - 3.3.5 Self Healing..... 26
- 4. Modules Technical Description 27**
 - 4.1 Sensor Failure Management 27**
 - 4.1.1 Tasks and Actions..... 27
 - 4.1.2 Protocol..... 28
 - 4.2 Failure Anticipation 31**
 - 4.2.1 Tasks and Actions..... 31
 - 4.2.2 Protocol..... 32
 - 4.3 Failure and Abnormal Behaviour Detection 33**
 - 4.3.1 Tasks and Actions..... 33
 - 4.3.2 Protocol..... 35

4.4 Resilience Backup37
 4.4.1 Tasks and Actions.....37
 4.4.2 Protocol.....38
4.5 Self Healing40
 4.5.1 Tasks and Actions.....40
 4.5.2 Protocol.....40
5. Conclusion.....42
6. Acknowledgement.....43
7. References44

LIST OF ABBREVIATIONS

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ABSR	Agent Based Self-Recovery
ADL-FEC	Adaptive Data Length-Forward Error Correction
AES	Advanced Encryption Standard
AFEC	Adaptive Forward Error Correction Control Code
ARQ	Automatic Repeat Request
ASE	Autonomic Sensor Element
BER	Bit Error Rate
CI	Critical Infrastructure
CN	Core Network
CRRM	Common Radio Resource Management
D3R	Decentralized Distributed Dynamic Routing
DA	Deciding Agent
DFT/HA	Distributed Fault-Tolerant / High Availability
DoS	Denial of Service
DSCP	Differentiated Services Code Point
EA	Executing Agent
FEC	Forward Error Correction
FP6/FP7	Sixth/Seventh Framework Programme
FRBS	Fuzzy Rule-Based System
FT/HA	Distributed Fault-Tolerant / High Availability
GNR	Go-Back-N
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPTV	Internet Protocol Television
IT	Information Technology
KB	Knowledge Base
MA	Monitoring Agent
MAC	Medium Access Control; Message Authentication Code
PIV	Program-Integrity Verification
PIVC	Program-Integrity Verification Code

PIVS	Program-Integrity Verification Server
QoE	Quality of Experience
QoS	Quality of Service
RF4CE	Radio Frequency for Consumer Equipment
RFC	Request For Comments
RSVP	Resource Reservation Protocol
SNR	Signal-to-Noise Ratio
SR	Selective Repeat
SW	Send and Wait
VM	Virtual Machine
WAN	Wide Area Network
WSAN	Wireless Sensor and Actor Network
WSN	Wireless Sensor Network

1. INTRODUCTION

This deliverable represents the outcome of TWISNet task 3.4 regarding communication, information and service availability. It addresses the technical themes of trustworthy assessment of system availability (failure anticipation, prevention, detection) and system resilience solutions, possibly in a multi-administrative domain environment.

To identify relevant architecture modules, the current state of the art is examined in chapter 2. All the relevant topics as mentioned in the section above are covered in the analysis. The actual literature is presented with respect to collaborative projects, existing standards and available commercial products. A summary concludes the various topics of the research.

Chapter 2 presents the architecture with a number of modules that compose the addressed framework, namely Sensor Failure Management, Failure Anticipation, Failure and Abnormal Behaviour Detection, Resilience Backup and Self Healing. The architecture shows the interactions between the identified modules within this framework and with other modules of the TWISNet framework.

A detailed description of these security modules is given in chapter 4. In particular, actions and tasks for each module are described to illustrate the modules' functionality and protocol for its realization.

Chapter 5 concludes with a summary and an outlook on further development.

2. STATE OF THE ART

2.1 Research literature

2.1.1 Trustworthy assessment

2.1.1.1 Remote monitoring of data integrity

The security and availability of sensor data should be guaranteed over the whole lifetime. However, data are subject to not only Byzantine failures, but also dynamic pollution attacks, as along the time the adversary may modify/pollute the stored data by compromising individual sensors. There are a number of protocols that have been proposed to remotely verify the integrity of the data or the programs residing at the sensor nodes.

With the scheme proposed in [1], data is not stored locally at node's memory for data dependability concerns; instead, data is distributed to other network nodes. The scheme enables the data owner with the capability to remotely check if its data still does exist in the network. Based on some metadata information, a verifier can periodically verify the integrity and the availability of the remote data an unrestricted number of times. The verifier can be either the data owner or some special fixed nodes, called "throwboxes", to whom the owner has delegated the verification of its data because for instance it has left the network. The proposed scheme is based on low-complexity computation operations; however, it achieves only probabilistic verification i.e., only the integrity of a portion of the stored data is checked at each verification operation.

The scheme in [2] allows deterministic verification. In this scheme, data along with integrity verification materials are coded based on an erasure code and secret sharing schemes, and then distributed to node's neighbors. The scheme proposes to add to the storage of a data share a parity block that is distinct for each share holder; thus preventing holders' collusion. The remotely stored shares are checked with data integrity verification routine exploiting the technique of algebraic signatures. Any data share holder can initiate the verification process by broadcasting a challenge message to the other share holders. The holders answer with signatures of their shares based on the challenge. The initiating holder can verify the integrity of the remote shares by comparing the parity of the received signatures with the signature of the stored parity. The proposed scheme is lightweight since transmitted signatures are just data symbols. The scheme can also support data maintenance, for instance a mobile sink can periodically collect data shares and parities and perform error correction when needed.

The Program-Integrity Verification (PIV) protocol in [3] is a software-based protection that allows remotely verifying the integrity of programs at sensor devices. PIV is triggered when a new sensor joins the network or when an existing sensor is removed from the network or if a sensor is suspected to have been compromised. In the proposed protocol, program verifiers, called PIVS (PIV servers), are distributed over the network and are in charge of verifying sensor programs. This role can be played by cluster-heads in a hierarchical WSN. A PIVS stores a database containing the digests of registered sensor programs. Thanks to the property that sensors generally share a portion of programs, the total number of distinct digests can be greatly reduced. A PIVS employs a mobile agent, called PIVC (PIV code), which is generated by the PIVS and executed at the sensor for verification. During a PIV, a PIVS remotely calculates through PIVC a random hash value of the program and another hash value using the digests it is storing, and then compares both values. The used randomized hash function achieves low processing and memory overhead; while allowing PIVC to be randomly generated whenever the program needs to be verified.

2.1.1.2 QoS parameters monitoring

QoS is a relatively new field in combination with Wireless Sensor Networks. Due to the resource restrictions of such sensor nodes, QoS is a challenging task. Since there a wide range of applications, one QoS for all types of application is not possible.

The QoS requirements are divided into guaranteed services (hard QoS) and differentiated services (soft QoS). Since different applications of a WSN require different QoS but use the data from just on sensor, the traditional end-to-end QoS parameters may not be sufficient to describe those differences. As a result, some new QoS parameters are desired for the measurement of the delivery of the sensor data in an efficient and effective way. To achieve QoS support in ad hoc networks, the literature specifies QoS model, QoS resource reservation signalling, QoS routing, and QoS Medium Access Control (MAC). A QoS model specifies an architecture and impacts the functionality of other QoS components. QoS signalling acts as a control centre, since it coordinates the behaviour of QoS routing, QoS MAC, and other components. The QoS routing process searches for a path with enough resources but does not reserve resources. To achieve QoS, in the literature two approaches are exploited, reservation-based and reservation-less. In latter approach, no reservation is required. While reservation-based approaches assign network resources according to an application's QoS [4].

In [5] two QoS perspectives are specified:

- **Application-specific perspective**
Focuses on the quality of the application itself, such as lifetime, coverage, deployment, quality of the sensing, camera resolution, number of active sensors.
- **Network-specific perspective**
Provides service quality during delivery of the data by the communication network.

Further, [5] specify resource constraints, node deployment, topology changes, data redundancy, multiple traffic types, real-time traffic, unbalanced traffic and scalability as the main QoS challenges in Wireless Sensor Networks. [5] also lists a number of differentiated MAC protocols and application specific protocols that take QoS support into account. These protocols differ from the numerous WSN MAC protocols existing in literature which focus on energy awareness.

2.1.2 System resilience solutions

2.1.2.1 Collaborative inter-domain failure anticipation, prevention and detection

2.1.2.1.1 Collaboration mechanisms of inter-domain NWK protocols in scope of failure anticipation

The research paper [6] presents a network and service failure restoration and prevention in multi-hop wireless and mobile networks. This QoS support mechanism is deployed in a heterogeneous wireless sensor network consisting of medical sensors and wireless gateways, called *Carenet*. This system bases on an intelligent QoS admission control mechanism, which is aware of the possible *states* that can be monitored (corresponding to plausible physical scenarios) and the consequences of these states on the sensors throughput. Meanwhile, the system is aware of the QoS requirements that correspond to these states (e.g. some emergency scenario may require high throughput because video monitoring would be necessary). In order to make in real time an admission decision, the worst case network utilization is considered.

While adopting a rather centralized than peer-to-peer approach, the solution presented in [7] provides an interesting approach for collaborative failure prevention in the sleep framework of sensor networks. The solution aims at achieving optimal redundancy versus lifetime trade-off for environment monitoring. It features two classes of nodes, namely "sensors" and "sentries", which are more powerful nodes. An adaptive sleep protocol is then designed, which works as follows: sentries first detect faulty nodes. They are then able to design sleep schemes for non-faulty ones in order both to act as a replacement for the faulty nodes, and to prevent failures on non-faulty nodes that might occur due to long 'ON' time.

2.1.2.1.2 Mechanisms on PHY+MAC layer protocols for failure anticipation

In [4] three basic mechanisms commonly used for error control are mentioned:

- **Automatic Repeat Request (ARQ)**
Define the persistent retransmission until data is successfully delivered. It can improve successful data delivery, but on the other hand increase latency, drop ration and energy consumption.
- **Forward Error Correction (FEC)**
In this technique, the sender encodes the data to be transmitted in a redundant manner, so that recipients can detect and correct transmission errors without asking the sender again. This mechanism brings extra latency caused by transmission of longer data packets. Also the algorithm needs extra computation power.
- **Hybrid ARQ**
It first sends out a data without (or minimal) FEC mechanisms. If any error occurs data is re-transmitted with use of FEC. This cycle continues until the packet is successfully delivered.

The classical ARQ protocols used for wired-networks use three basic schemes:

- **Send and Wait (SW):**
Transmitter waits for an ACK or timeout before next transmission
- **Go-Back-N (GNR):**
Transmitter sends continuously which number is adjusted with use of a sliding window. If any packet failed all packets within the sliding window re-transmitted.
- **Selective Repeat (SR):**
Transmitter sends continuously which number is adjusted with use of a sliding window. If any packet failed, just the packet that failed is re-transmitted.

In [8] the following adaptations of these schemes are presented:

- **Implicit Acknowledgement**
All packets are sent in broadcast mode and overhearing behavior is used. Sensor nodes listen to packets send by neighbors, which in turn send the packet on the way to the sink. This can be seen as an hop-by-hop acknowledge.
- **Selective Acknowledgment**
Packets are divided into critical and non-critical packets. Critical packets should be acknowledged. The easiest way to define if a packet is critical or not is to agree on a threshold between the sink and the sensor nodes before deployment. If the sink

receives a critical packet it send a acknowledge packet immediately back to source node.

- **Enforced Acknowledgment**

It follows the same scheme as Selective Acknowledge, just that the source node decides whether the packet is critical or not (without threshold) and marks the packet accordingly. The sink then acknowledges marked packets.

- **Blanket Acknowledgment**

The idea is that multiple sensors in a WSN which consists of thousands of nodes, are reporting the same event. This event should just be acknowledged by a single acknowledgement packet.

In [9] an Adaptive FEC control code (AFEC) is presented which dynamically changes the amount of FEC code, based on the number of received acknowledgements for its packets. They don't take any specific information about SNR or BER from the receiver into account. In [10] an Adaptive Data Length-Forward Error Correction (ADL-FEC) mechanism is presented which chooses, depending on the wireless channel status, an appropriate FEC code to correct damaged packets.

In [4] maximizing throughput, minimizing delay, minimizing delay jitter and maximizing energy efficiency are listed as general metrics for a networking perspective. MAC specific metrics are:

- Minimizing medium access delay
- Minimizing collisions
- Maximizing reliability
- Minimizing energy consumption
- Minimizing interference and maximizing concurrency
- Maximizing adaptivity to changes

All this facts should be taken into account when thinking about mechanisms on MAC layer protocols. There are many MAC protocols placed on different applications, which take care of this metrics. A good overview about all the different MAC protocols can be found in [5],[11],[12],[13] and [14].

The IEEE 802.15.4 standard [15] also specifies a beacon-enabled transmission mode, which offers a guaranteed time slot technique, where sensor nodes can obtain a free spot and send messages contention-free.

2.1.2.1.3 Detection of failures and abnormal behaviour

Anomalous output recognition

WSANs have many characteristics that make them different from previously studied computing environments. In terms of reliability in the presence of attacks, several factors make this especially challenging for WSANs. On the one hand, WSANs are particularly susceptible to attacks. Sensor nodes, by definition, are designed to interact with the environment and are therefore susceptible to attack from it. The nodes will also need to be low-cost, and therefore will be lower quality and more unreliable and susceptible to attack.

On the other hand, there are significant constraints in any intrusion detection solution that is applied to WSANs. They may be deployed in large numbers and over widespread areas making it difficult to physically access them in order to diagnose and correct problems using procedural solutions. The nodes themselves are likely to be highly resource constrained, in terms of energy, memory and processing power, and therefore technical solutions with low overhead will be required.

As attacks cannot be prevented, what is needed is a lightweight way of detecting problems with the data output by the sensors in the form of some sort of "plausibility check". Detection is our main focus; however, ideally, the solution would also be able to locate and diagnose the problem as well as enable some kind of corrective actions to take place. In this section we survey the state-of-the-art that may be applicable to this problem.

Fault detection

Fault management has a long history within computing, and many tried and tested existing models, such as the use of redundancy (e.g. majority voting and the "Byzantine generals' problem") and diversity (i.e. using multiple distinct implementations for the same task). Applying such techniques directly to WSANs is not completely straightforward due to their peculiar characteristics and constraints as mentioned above, and is a relatively new area of research. Some survey papers of work that has been done in this area include [16] and, more recently [17]. In [18], taxonomy of approaches to fault management is described. This is summarised below.

Centralised approaches collect information at a central point (e.g. a gateway) at which an analysis of the whole network can take place, as well as selective probing of nodes. The main problem with this approach is the additional message overhead of sending information to the centre, as well as the fact that the centre is a single point of failure and nodes near to it can suffer from premature energy depletion due to the increased load of transferring messages to and from it. Delays in detecting and reacting to faults can also be a problem. However, this approach can be good at detecting faults and is generally the most accurate.

Distributed approaches rely on nodes within the network detecting faults. In some approaches, nodes perform self-detection of faults, but this is chiefly used for hardware fault detection on nodes themselves. Neighbourhood-based approaches make use of coordination with neighbouring nodes. For example, nodes can compare readings to detect anomalies (e.g. the Byzantine Generals Problem). In other approaches, "watchdog" or observer nodes monitor their neighbouring nodes for how they deal with the routing of traffic. Distributed approaches can detect problems early on and do not suffer the same message overheads as centralised approaches. However, they suffer from the restricted resources available on the sensor nodes, which means that in practice, determination of faults can be slow and the overhead on nodes can lead to premature energy depletion and limits the amount of monitoring that can actually take place.

Cluster-based or hierarchical approaches provide an intermediate approach between distributed and centralised approaches by dividing up the WSAN into regions or "clusters", with clusterheads being responsible for monitoring their own clusters and coordinating between clusterheads.

Fault Diagnosis

Fault diagnosis depends on the fault model used within individual approaches, which is nearly always application-specific. Currently, there is no common or comprehensive fault model defined for WSANs. One problem is that the fault model may differ significantly between scenarios, and therefore a flexible approach to fault diagnosis may be required. A similar problem exists when considering intrusions, as the selection of the threat model will have a significant impact on the design of the intrusion management approach.

Fault Recovery

Generally, the approach taken is to isolate misbehaving nodes by modifying the routing tables on nodes. Some work on the reconfiguration or reprogramming of nodes may be relevant, particularly for fault correction but less so for compromised nodes.

Joaquin et al. [19] combines collaborative knowledge with WSNs. For demonstrating his concept, he implements a collaborative knowledge based system that tracks the evolution of the behaviour of the olive tree pets based on the sensors measurements (humidity and temperature). The knowledge is shared among the sensors; in each sensor an extension of the rule-based system, called Fuzzy Rule-Based System (FRBS) [20] is adapted. Sharing the data, knowledge and variables, the sensors improve their decisions and cope better with errors and uncertain data.

2.1.2.2 Actions in case of failure detection

2.1.2.2.1 Service survivability and recovery

Service survivability in case of failure detection has been addressed in literature mainly by constructing network structures and topologies that are well suited for resilience and fault tolerance.

Paradis and Han [16] presented a survey about fault management in wireless sensor networks. In order to be able to survive and to recover from failures, they identified the core sources of faults. They consist of:

- Network congestion: Congested networks affect the availability and the performance of networks.
- Battery depletion: Wireless sensor networks are highly constrained by their energy consumption. Battery depletion will occur either if the sensor network is not properly maintained or by external attackers that aim at compromising availability or confidentiality.
- Environmental influences: Depending on their field of application, sensors might be exposed directly to rough environments. For example weather conditions could affect the availability of nodes.

Those factors all influence the capability to survive or to recover from failures in the networks. Additionally, multi-hop networks increase the need for sophisticated network structures that are resilient against unintended network behavior. The authors further identify fault avoidance, fault detection, fault isolation and finally fault recovery as the sequential steps how to deal with abnormal network behavior.

Gupta et al. [21] present a structural method to make the network able to recover from failures. They divide the sensor network into clusters, where each cluster contains higher-powered “command” nodes. Having dedicated nodes that are doing energy-demanding computation and communication decreases the burden to all other normal nodes and prevent unexpected battery depletion. The mechanism presented by the authors allows nodes to quit failed clusters and to join another, healthy one. The main advantage is that no redundant gateways are required for the case that one node failed. Furthermore no full re-clustering has to be done. Failed nodes are identified when their delivered data deviates from the specification. Failure detection is done by consensus. After a failure has been detected, the nodes agree to join new clusters and new TDMA schedules are negotiated.

A mechanism that addresses availability of the network in case of network congestion is Coda presented by Wan [22]. The idea is to send hop-by-hop “backpressure” messages in case of network congestion. Nodes on the hop-by-hop path will adjust their sending rate

accordingly to mitigate congestion. Sink nodes will suppress ACK messages to indicate to the sources to reduce their sending rate.

A more general view on survivability of wireless networks is given by Quian et al. [23]. As special survivability requirements they identify reliability, availability and energy-efficiency. As key factor they identify a flexible key management in the WSN. The drawback of more flexible key management is a loss of security since more nodes need to have knowledge of certain keys to enable survivable service. The authors show that a security-survivability tradeoff is inevitable.

2.1.2.2.2 Self-healing mechanisms

An autonomic system is composed of interrelated autonomic elements. Each of these elements has managed hardware or software resources that build the IT infrastructure and autonomic managers that supervise and control these resources. Self-healing components can detect system malfunctions or failures and start corrective actions based on defined policies to recover the network or a node. The automatic recovery from damages improves the service availability.

As described in [24],[25],[26],[18] there are different healing strategies based on locality awareness of the sensor node and/or energy efficient algorithms.

Graph heal: On each deletion, the neighbours of the deleted node in a binary tree are reconnected regardless of whether any cycles in the graph formed by the new edges introduced for healing. The downside of this algorithm is that the nodes use more edges than what is required for maintaining connectivity.

Binary tree heal: [27] On each deletion, the neighbours of the deleted node in a binary tree being are reconnected careful not to introduce any cycles in the graph formed by the new edges introduced for healing. This is done using random IDs which can then be used to identify which tree a particular node belongs to. This is an improvement on the previous algorithm but still naive since it does not take into consideration the previous degree increase suffered by nodes during healing.

Automatic Fault Recognition: [28] Rather than having a system response based on node deletion events, another self-healing approach is trying to detect fault patterns and have so-called B-scripts react and repair the fault. This strategy defines two new node types: lymph and thymus that detect anomalies and malicious behaviour and submit corrective commands to the system.

The basic assumption of all current algorithms is that the network is initially a connected graph over n nodes. An adversary repeatedly attacks the network. This adversary knows the network topology and the algorithms, and it has the ability to delete arbitrary nodes from the network. However, we assume the adversary is constrained in that in any time step it can only delete a single node from the network. Another assumption is that after the adversary deletes some node x from the network, that the neighbours of x become aware of this deletion and that the network has a small amount of time to react by adding and deleting some edges.

Several algorithms implement these strategies, the most efficient being:

The Forgiving Tree [25] is based on a rooted spanning tree T -layout, which without loss of generality may as well be the entire network. Each time a non-leaf node v is deleted, it is replaced by a balanced binary tree of „virtual nodes” with the leaves of the virtual tree taking v 's place as the parents of v 's children.

When a leaf node is deleted, it is not replaced. However, if the parent of the deleted leaf node was a virtual node, its degree has now reduced from 3 to 2, at which point it is

considered redundant and „short-circuited”, removing it from the graph, and connecting its surviving child directly to its parent. This helps to ensure that, except for heirs, every virtual node is of degree exactly 3. The replacement of each deleted node by its virtual tree can be done in $O(1)$ time. The total size and number of messages which must be sent is $O(1)$ per deleted vertex. In addition, there is a startup cost for communicating the initial “wills”; this is $O(1)$ per edge in the original network.

DASH [27] improves on [25] on several aspects, one of which being that no node ever increases its degree by more than $2 \log n$, where n is the number of nodes initially in the network. DASH adds new edges only among neighbours of deleted nodes; and has average latency and bandwidth costs that are at most logarithmic in n . DASH has these properties irrespective of the topology of the initial network, and is thus orthogonal and complementary to traditional topology-based approaches to defending against attack.

When a node x is deleted, the neighbours of x react to this deletion by adding some set of edges amongst themselves. These edges can only be between nodes which were previously neighbours of x . This is to ensure that, as much as possible, edges are added which respect locality information in the underlying network.

DASH is a fully distributed algorithm and it also has several “flavours” like **SDASH** (Surrogate degree based binary tree heal), a heuristic algorithm that tries to keep both node degrees and path lengths small.

SASHA [28] is a self-healing hybrid sensor network architecture that is inspired by and co-opts several mechanisms from the acquired Natural Immune System to attain its autonomy, and adaptability to unknown faults. SASHA encompasses automatic fault recognition and response over a wide range of possible faults and tries to define an adaptive architecture that can learn and evolve its monitoring and inference capabilities over time to deal with unknown faults. Two new node types are defined: the *lymph* tries to detect network anomalies and the *thymus*. In a WSN the survey of a forest can be undertaken by means of mobile scripts running on all monitors, called B-script. A script is dynamically generated code and it acts as a filter for the behaviour and statistical analysis of a forest. The second node type is the *thymus* that tries to store a representation of the current system status known as self and confirm the presence of faults.

This approach is different than the rest of the standard algorithms and more complex. SASHA proposes algorithms for automatic fault recognition, adaptive network monitor and coordinated response. Although there is no working implementation of the algorithm, this self-healing strategy could base itself on other services already described in this document like code migration, sensor code update and in-network computation and could also provide better overall system response.

2.1.2.2.3 Dynamic role management on device level

On device level, various faults and attacks can interfere with normal functionality. They should be identified and mitigated by security mechanisms.

Sastry et al. evaluate the security of IEEE 802.15.4 networks [29]. They determine that the optional features reduce security instead of enhancing it and can introduce vulnerabilities that will not be detected once the network is deployed. They describe the way the specification can be changed in order to prevent those vulnerabilities.

Raymond and Midkiff present the Denial of Service attacks against sensor networks and describe some of the defences [30]. At the physical layer the DoS attacks are jamming, node tampering and destruction; at the link layer are interrogation and Denial of Sleep attack; at the network layer are spoofing, replaying, altering routing control traffic or clustering messages, Hello floods and homing attacks; at the transport layer the SYN flood and de-

synchronization attacks; at the application layer the overwhelming of sensor nodes, the path-based DoS and the Deluge attack.

Raymond et al. describe the effects of the Denial of Sleep attacks on the MAC protocol in WSNs [31]. The Denial of Sleep attack targets the node's power supply in order to reduce its lifetime. The authors classify the Denial of Sleep attacks and determine the impact on the sensor network of each attack. They also introduce a framework that is able to prevent Denial of Sleep attacks.

FT-CoWiseNets is a Fault Tolerance Framework for WSNs that has been developed by Souza [32]. The causes of faults can be the malfunctioning hardware, the software errors or other external causes. In sensor networks, faults must be detected and the devices should be able to recover from them automatically. For this purpose, the authors developed FT-CoWiseNets, which is a framework that is able to provide fault-tolerance to heterogeneous sensor networks by including fault diagnosis and recovery techniques.

Ishikawa et al. developed a Framework for Self-healing Device Drivers [33]. Most of the times, device drivers cause failures in the operating systems. The authors designed a framework that adds a self-healing ability to device drivers and permits developers to write code for failure recovery for their device drivers. A prototype has been implemented by the authors on top of L4 microkernel and it was able to fully recover from crash in just 0.2 ms.

Li et al. propose a mechanism for Fast Recovery from Node Compromise in WSNs [34]. This mechanism relies on the dynamic reclustering of the network and on node reprogramming. The network reclustering mechanism is used to exclude compromised sensor nodes from the network in order to enable normal operation while the node is recovering. The clustering algorithm uses the energy level of each sensor node and the connectivity to other nodes in order to compute the metric. The recovery procedure consists in reprogramming the node using the Deluge protocol.

ABSR is an Agent based Self-Recovery Model for WSNs developed by Ma et al. [35]. The model uses automatic computing technology in order to perform anomaly monitoring and detection and after that, sensor node recovery. The compromised sensor nodes are monitored cooperatively by the agents, which perform operations to recover the node. Autonomic sensor element (ASE) based on agents runs on each cluster head and base station, which have more resources than the other sensor nodes. ASE includes the monitoring agent (MA), the deciding agent (DA), the executing agent (EA) and the knowledge base (KB), each having specific roles in the detection and recovery processes.

2.2 Collaborative projects

2.2.1 Trustworthy assessment

The "Mechanisms for Optimization of hybrid ad-hoc networks and satellite NETWORKS" (MONET) is a FP7 project that searches new approaches with hybrid MANET-Satellite networks. This combination raises significant challenges in terms of optimising network resources, link availability, providing Quality of Service (QoS) and Quality of Experience (QoE) and minimizing costs and energy consumption. Issues such as the re-organisation of MANET to connect to satellite access points, re-organisation of the satellite access points, selection of which satellite access points to use, the use of satellite as a relay between two MANET, the adjustment of routing in accordance with the current network situation and the exchange of cross layer information to improve resource management will be investigated in MONET [36].

The "Enhanced, ubiquitous, and dependable broadband access using MESH Networks" (EU-MESH) is a FP7 project that's goal is to develop, evaluate, and trial a system of software

modules for building dependable multi-radio multi-channel mesh networks with QoS support that provide ubiquitous and ultra-high speed broadband access [37].

The “Policy Based Management of Heterogeneous Networks for Guaranteed QoS” (NETQOS) is a FP6 project with the main objective of the development of an autonomous policy-based QoS management for wired/wireless heterogeneous communications networks aimed to provide enhanced end-to-end QoS and efficient resource utilisation [38].

The “Advanced resource management solutions for future all IP heterogeneous mobile Radio Environments” (AROMA) is a FP6 project that main objective is not only to assess and maximize the potential benefits coming from the medium-term evolution of the considered radio-access technologies but in parallel also to promote and investigate potential benefits coming from a long-term evolution towards an all IP heterogeneous mobile and wireless network architecture. In order to support end-to-end QoS in a heterogeneous wired and wireless mobile environment, an appropriate interaction between the QoS management entities of the core network (CN) and the Common Radio Resource Management (CRRM) in the radio part is crucial [39].

2.2.2 System resilience solutions

The ANA (Autonomic Network Architecture) project [40] is an EU-funded FP6 project that aimed at exploring novel ways of organizing and using networks beyond legacy Internet technology. Especially, ANA designed and developed a novel autonomic network architecture enabling flexible, dynamic, and fully autonomous formation of network nodes as well as whole networks. The "resilience overlay compartment" is the unit within which nodes collaborate with each other in order to facilitate early anomaly prevention.

SENSEI (Integrating the Physical with the Digital World of the Network of the Future) is a FP7 Integrated Project [41], finished in December 2010. SENSEI creates an open, business driven architecture that fundamentally addresses the scalability problems for a large number of globally distributed WSN devices. It provides the necessary network and information management services to enable reliable and accurate context information retrieval and interaction with the physical environment. By adding mechanisms for accounting, security, privacy and trust it enables an open and secure market space for context-awareness and real world interaction. In the context of state-of-the-art WSN island solutions the project provides advanced detection mechanisms for abnormal behaviour in the form of outlier detection algorithms.

WSAN4CIP is a recently finished Specific Targeted Research Project (STREP), which is partly funded by the European Commission in the ICT security area of the Seventh Framework Programme (FP7) under Objective 1.7 "Critical Infrastructure Protection" [42]. The goal of WSAN4CIP is to advance the technology of Wireless Sensor and Actuator Networks (WSANs) beyond the current state of the art, in order to improve the protection of Critical Infrastructures (CIs). By advancing WSN technology, the project contributed to networked information and process control systems which are more secure and resilient. The distributed nature of WSANs enables them to survive malicious attacks as well as accidents and operational failures. It makes them dependable in critical situations, when information is needed to prevent further damage to CIs. The project will help enhance the reliability of critical infrastructures by providing surveillance data for the management of the CI. It will increase the dependability of critical infrastructures security, by providing self-healing and dependability modules for the WSN. Also it will provide a dependability engineering methodology and appropriate tool support. Finally, it will demonstrate the feasibility of the project's technical approach by using energy generation and distribution as an example of a critical infrastructure.

TAMPer REsistant Sensor node (TAMPRES) is a Collaborative Project supported by the European Commission in the context of the Seventh Framework Programme under the "Trustworthy ICT" sub-programme area that focuses on improving the trustworthiness of WSNs [43]. The project includes prevention of side-channel and fault-injection attacks, the implementation of light-weight cryptographic cores and the development of attack resistant system architecture. The project aims to provide tamper resistance solutions for low-power devices, while maintain their cost as low as possible. The attack resistant architecture should provide secure system integration and partitioning and includes secure low level services as well as hardware components.

INcreasing Security and Protection through Infrastructure Resilience (INSPIRE) is a Collaborative Project supported by the European Commission in the context of the Seventh Framework Programme that was finalised in 2010 [44]. The project aimed to identify methods to improve the communication reliability when dealing with unreliable or insecure links such as WAN or wireless. They wanted to provide a method that identifies and evaluates vulnerabilities and techniques for attack and failure evaluation and recovery. They focused on developing a framework that is able to deal with multiple types of faults and attacks, which includes a fault/intrusion tolerance manager that chooses the most appropriate recovery strategy and applies it.

INfrastructure for heTERogeneous, Resilient, SEcure, Complex, Tightly Inter-Operating Networks (INTERSECTION) is a Collaborative Project co-funded by the European Commission in the context of the Seventh Framework Programme under the "Secure, Dependable and Trusted Infrastructures" sub-programme area that has been finalized in 2009 [45]. The project aimed to provide resilient methodologies and architectures that are able to indentify, resist and autonomously respond to events that are anomalous. They wanted to develop new and efficient intrusion detection and tolerance techniques. The system should detect well-known or new types of anomalous events and react to them.

Assessing, Measuring, and BEnchmarking Resilience (AMBER) is a Collaborative Project co-funded by the European Commission in the context of the Seventh Framework Programme under the "Secure, Dependable and Trusted Infrastructures" sub-programme area that focuses on measuring resilience and benchmarking for computer systems and components [46]. They aim to provide metrics definition and develop benchmarks that are able to evaluate the resilience of systems and components.

2.3 Standards

2.3.1 Trustworthy assessment

IntServ (Integrated Services) is a fine-grained Quality of Service (QoS) technique to prioritize IP data packets. In contrast to the DiffServ method, the resources for individual links and not for traffic classes are required. The idea of IntServ is that every router in the system implements IntServ, and every application that requires some kind of guarantees has to make an individual reservation. To construct resource reservations, the Resource Reservation Protocol (RSVP) is used. The problem of IntServ is that many states must be stored in each router and so when the system is scaled upwards, it becomes difficult to keep track of all of the reservations. As a result, IntServ is not very popular. An overview about IntServ protocol is given in RFC 1633 [47].

DiffServ (Differentiated Services) is a framework for the classification of IP packets. This classification can be used to prioritize IP data packets to guarantee a QoS. DiffServ was first described in RFC 2474 in 1998 [48]. DiffServ uses the 6-bit Differentiated Services Code Point (DSCP) field in the IP header for packet classification purposes. DSCP replaces the outdated Type of Service field.

The IEEE802.11e-2005 [49] is an approved amendment to the IEEE 802.11 standard that defines a set of Quality of Service enhancements for wireless LAN applications through modifications to the Media Access Control (MAC) layer.

The HomePNA Alliance is a non-profit industry association of companies that develops and standardizes technology for home networking over the existing coaxial cables and telephone wiring within the home. With the HomePNA version 3.1 [50] they added features such as guaranteed Quality of Service for such applications as IPTV which require consistent high performance over the entire house.

2.3.2 System resilience solutions

The IEEE 802.15.4 standard includes several security features such as that provide authentication, integrity, confidentiality and access control in order to protect against malicious attacks [15]. The encryption algorithm used in this standard is AES with 128 bits keys. The authentication and integrity is provided through the Message Authentication Code (MAC) that can be on 32, 64 and 128 bits and is computed using the message and the secret key based in the 128 bits AES mechanism. The standard enables the use of 8 security levels which allows for security to be adapted to each scenario and its requirements. Access Control Lists are introduced by the standard; the list and the security policy can be used to control which nodes should be trusted (trusted brothers). The node will send packets only to trusted brothers and it will deny any message from nodes that are not trusted until they have been authenticated.

RFC 816 deals with Fault Isolation and Recovery [51]. The action in which hosts and gateways collaborate to take a decision that something wrong has happened is called fault isolation. Fault recovery deals with the identification and the selection of an alternative route from source to destination in the case a fault on the initial path is discovered. The gateways should be able to always provide a correct and consistent model of the internetwork topology. The host nodes can interrogate the gateway whenever they detect a suspicious event.

2.4 Commercial products

Arch Rock is one of the first has one of the first commercial implementations of the IETF 6LoWPAN, Primer Pack/IP [52][52]. Their technology has been incorporated in the Advanced Incident Report System, a network that allows personnel from various local, municipal and governmental agencies to communicate during an event, making reliability one of their main goals. In terms of detecting failures and abnormal behaviour, the Arch Rock sensor nodes send heartbeat packets periodically to check network reliability and connectivity. The packets contain information that the management server can analyse in order to detect any problems, such as node health statistics. Due to the use of native IP, standard network management tools, such as sniffers, can be used to diagnose problems.

In the SmartMesh networks from the company Dust Networks, a manager has a key role in applying routing algorithms in a self-configurable network [53], in order to optimize paths, being capable of both detecting problems and optimizing functionality.

A lot of the commercial applications boast on their reliability. Some of them mention self-healing as an attribute. Dust Networks uses full mesh self-configuring WSNs in their SmartMesh networks [53] to provide flexibility, while a network manager uses routing algorithms to continually optimize the routing of data through the network.

The Meshscape 5 System from Millennial Net using an algorithm called D3R (Decentralized Distributed Dynamic Routing) which enables nodes to repair broken links and allows for frequency hopping [54]. BBN technology is used in military applications which emphasises

the reliability requirement, so they dropped the dynamic hierarchy for routing in favour of full mesh networks with equal peers [52]. Green Peak boast on their RF4CE chips to offer an additional 30db (1000x) better interference robustness due to their patented technology of two separate antennas [55].

The VMWare Fault Tolerance (FT) provides hardware style fault tolerance to virtual machines by using the encapsulation properties of virtualization to assure high availability into the x86 hypervisor [56]. The VMWare vLockstep is used to manage an active secondary virtual machine which runs in virtual lockstep with the primary one. The secondary virtual machine will reside on a distinct computing system and will execute the same sequence of virtual instructions as the primary. The same inputs can be observed by the secondary and should be able to become primary without service interruption or loss when the fail of the primary virtual machine is detected. Therefore, it provides uninterrupted availability in the presence of critical faults, even when the primary machine is down.

Trillium provides innovative protocol software solutions that enable Fault-Tolerant / High Availability (FT/HA) and Distributed Fault-Tolerant / High Availability (DFT/HA) [57]. They developed a flexible, platform-independent and cost-effective framework that is able to provide connectivity in the presence of software and hardware failures. It distributes the protocol load on the physical processing units that are available and then dynamically re-distributes them in the case of processor failure or when a new processor is added in the system. It is able to recover from processor failure and allows maintenance without shutting down the system.

2.5 Summary

The proposed schemes for data monitoring generally assume that the verifier holds some information that allows the integrity verification of the remote data. Such information may consist of the actual data being verified [3] or credentials (e.g., public keys, parity checks) authenticating the remote data (e.g., [1], [2]). Aggregation of data monitoring operations is difficult to realize in this setting, since the proposed schemes generally rely on challenge-response messages for verification that may pass through multiple nodes holding the same data. Compromised nodes may then collude to keep only one data copy; while answering correctly to all challenges.

Due to their low cost, Wireless Sensor Networks are exceptionally susceptible to node failure and malfunctions. This is why management functionalities must be implemented in order to harness the adaptability of WSN islands. Existing network management solutions focus mainly on providing network connectivity but neglect other important parameters such as node status and coding integrity.

A better approach would be implementing Service-Level self healing mechanisms in the WSN islands. Each node knows about a larger set of tasks to execute and in case one node fails, the failed service will be rescheduled on a new node without user intervention. The change should be transparent and will support service addition and removal depending on network and node status.

Available literature concludes that low-power devices are vulnerable to faults and attacks and should be protected using security and fault-tolerance mechanisms. Raymond et al. investigated Denial of Service attacks and proposed defence methods against them. FT-CoWiseNets is a framework that provides fault detection and recovery. Multiple recovery and self-healing solutions are described in the available literature.

Several projects supported by the European Commission in the context of the Seventh Framework Programme, such as TAMPRES, INSPIRE, INTERSECTION and AMBER deal with resilience against faults and attacks.

IEEE 802.15.4 standard includes features for protecting devices against attacks. RFC 816 provides the specification of fault detection and recovery mechanisms.

The commercial product VMWare FT was designed for providing hardware style fault tolerance to virtual machines. The solution provided by Trillium provides fault tolerance and high availability. Commercial products that provide protection against faults are not available for WSNs.

Concluded, a number of solutions exist which focus on certain parts of security aspects. A complete resilient and fault tolerant system integrating state-of-the-art mechanisms and filling the gaps with innovative techniques is to be developed by TWISNet project.

3. ARCHITECTURE FUNCTIONAL DESCRIPTION

3.1 Overview

Purpose of this architecture is to ensure resilience and availability by direct security approaches but also by alternative approaches such as ensuring battery, fault-tolerance and reliability.

Particular interest is given to failures caused by physical damages or compromised security. A service quality assessment system employing a resilience server is defined which contains several modules to monitor availability of different service components and pinpoint unsecure networks. It also takes failure anticipation, prevention and detection into account. To achieve this, techniques for predicting failures that are likely to happen, detecting failures, and taking appropriate preventive actions (e.g. data replication, redundant logical entity etc.) are defined. Further techniques for detection of abnormal logical entity behaviour are also considered.

The whole failure management system is based on trusted entity modules which ensure effective failure detection and recovery. This system also includes a secure and trusted recovery system. In addition, the service availability problem is addressed also for the scenario where the service information flows across heterogeneous administrative domains or from a private to a public domain and vice versa. Techniques for setting up backup options (e.g., backup trusted entities, routing paths) are considered when service continuity cannot be ensured because of access limitations to some network infrastructures.

3.2 Presentation

The architecture covers a set of sensors, a resilience server and an application server as shown in Figure 1. It is divided into a number of modules dedicated to certain sub-tasks of the resilience system. The proposed architecture modules are described in the following section.

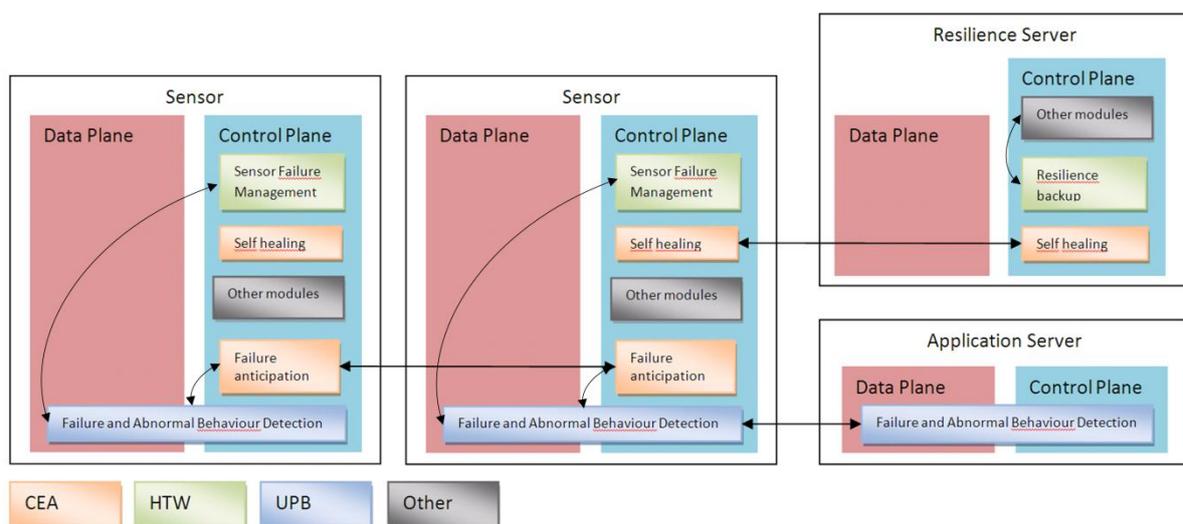


Figure 1: Overview of the architecture for resilience and availability

3.3 Modules Functional Description

3.3.1 Sensor Failure Management

The Sensor Failure Management module (SFM) is a sensor-local generic module, linking parameter values and computed results to according actions. It can be configured to monitor the sensor device locally and to use algorithms to detect failures that take place at the device level. In case of failure, recovery methods can be employed to eliminate the failure and its effects. For example, the Sensor Failure Management module can send a recovery request to the local Resilience Backup module which contacts the resilience server for further recovery procedures. For monitoring, different preconfigured modules can be optionally integrated or left out based on the scope of monitoring and the energy saving level that is required. SFM allows pull as well as push operation, meaning that actions are applied based on an own schedule as well as external modules are able to trigger local actions applied by SFM. For server notification, SFM is enabled to communicate errors through the Failure and Abnormal Behavior Detection (FABD) module.

3.3.2 Failure Anticipation

The failure anticipation module works on a peer-to-peer basis by interacting with similar modules on other sensor nodes.

The first objective of this module is to evaluate the local risk of failure of a node. For that purpose, there is no need for interconnecting with other peers. Rather, this local process takes into account past actions (e.g. consumed energy in the past hours) and induces whether their expected continuation is compatible with device characteristics (e.g. battery level). If this is not the case, an alert is raised that triggers both human intervention and intervention of the resilience system.

The second objective of this module is to perform collaborative actions with its peers in order to achieve global failure anticipation. This may consist for the module to report the min/mean/max time for a certain process to happen. Eventually a total delay can be induced as the sum of individual ones, and compared to acceptable ones in order to prevent unacceptable latencies. Global failure anticipation is also intended to prevent domino effects: by evaluating their respective dependencies and the available fallback solutions, nodes should be able to anticipate chain reactions and to identify single point of failures. A criticality metric can be derived during this process, wherein a node interacting with many peers that do not have a fallback solution and for essential matter would be assigned a very high metric value. Consecutively, a notification can be sent to either human operator or the resilience system.

The Failure Anticipation module is an optional module of the TWISNet framework.

3.3.3 Failure and Abnormal Behaviour Detection

The Failure and Abnormal Behaviour Detection module (FABD) monitors the sensor devices for abnormal output in order to detect node failure or the failure of neighbours. The nodes will communicate with the Application Server to compare their outcome with the information stored in a Data Plane module. In case of an intrusion, the FABD module of the Application Server will apply corrective methods. In this respect, requests to take preventive or recovery actions are sent to the appropriate modules of the sensor (the Secure Routing and the Sensor Failure Management modules).

3.3.4 Resilience Backup

The Resilience Backup module (RB) is responsible for managing schedules for creation and transfer of the partial configuration or full backups of the node's configuration values to a Resilience Database and for recovery from backups in case of failure. Backup is done periodically or on request. Therefore, RB provides a service for requesting a recovery, which may be generated by local or server failure management module and sent to the Resilience Backup module on the Application Server. The Resilience Backup modules need to interface with Secure Mechanisms for Device Reconfiguration (SMDR) and optionally with Firmware management for backup and recovery purposes and relies on the database managed by SMDR.

3.3.5 Self Healing

The Self Healing module (SH) in a sensor node collaborates with other similar modules within other nodes in order to enable a collaborative resilience system between nodes. Through this module, nodes exchange information relative to their needs and capabilities in terms of security. This is especially relevant when, because of some unanticipated failure, a security function offered by a node A to a node B stops being available. The Self Healing module comes then into action allowing B to locally discover another node, or a combination of other nodes, which can provide it with the same service.

This module relies on a local database for storing node's needs and capabilities.

The Self Healing module is an optional module of the TWISNet framework.

4. MODULES TECHNICAL DESCRIPTION

4.1 Sensor Failure Management

4.1.1 Tasks and Actions

The Sensor Failure Management module is a sensor-local generic module, linking parameter values and metrics to according actions. It can be configured to monitor the sensor device locally and to use algorithms to detect failures that take place at the device level.

Local parameters may be monitored on a pre-configured schedule or on request. By combining several parameters, a computed result, referred as metric, can be used. A parameter or metric may be linked to a set of thresholds. These combinations are linked to certain pre-configured actions.

The SFM module provides the generic framework for linking these subjects as well as a set of pre-configured function blocks which may be combined for local monitoring purposes. Therefore, in case of failure, recovery methods can be employed to eliminate the failure and its effects. For example, the Sensor Failure Management module can send a recovery request to the local Resilience Backup module which contacts the resilience server for further recovery procedures.

For monitoring, different preconfigured modules can be optionally integrated or left out based on the scope of monitoring and the energy saving level that is required. SFM allows pull as well as push operation, meaning that actions are applied based on an own schedule as well as external modules are able to trigger local actions applied by SFM. For server notification, SFM is enabled to communicate errors through the FABD module.

SFM may be used to detect internal risky situations related to the nodes' lifetime or functionality, like low battery situation and high workload by the definition of according metrics. This includes monitoring of energy status with respect to current and averaged workload. The system may be extended by other self-tests, i.e. if external components are used which support this kind of monitoring. The information is processed in configurable intervals or on dedicated pushed request, so metrics are calculated and thresholds are defined which lead to appropriate actions.

Also security risks may be monitored, for example by monitoring changes of critical configuration parts which are checked in regular intervals. By defining sets, context-aware capabilities of the node are exploited, i.e., physical harm may be detected if the node supports it. For example, if a node has the ability to detect when its case is opened, it can send a security failure message to FABD which is recognized at the application server. Such measurements can also be done indirectly if the context is clear, i.e. a light sensor can be used to detect if a case is opened.

Actions may include informing the network of an emergency status by using messages to FABD and requesting backup or recovery of certain parameters using the RB module. Pre-defined functional blocks may be extended or new ones may be written to be integrated into the generic SFM structure.

For example, emergency profiles may define to set the node into a state where functionality and security are supported at minimum level until the failure that triggered the event is resolved. An energy emergency profile may contain the parameters of the node which guarantees the longest lifetime possible whereas the basic functionality and security remains

working. In this way, a node is recognized to need maintenance at the application server through the FABD whereas the node itself continues working with limited functionality.

In scenarios where a large number of battery powered nodes have to perform self-check to request maintenance only like it is the case for industrial monitoring, security failure detection may be disabled to save some energy.

In supply and demand optimization, energy monitoring may be of less importance as the smart meter is probably connected to a mains powered supply. Nevertheless, security monitoring is more important here as the device is installed in an environment where it may be subject to physical or security attacks.

4.1.2 Protocol

SFM stores independent configurable sets for any information it has to monitor resp. metric it has to calculate which allows easy extension to further objects to be monitored. Single monitoring options may be disabled by changing the according set. A set may either be a parameter set, querying single parameters, or a metric set, querying other parameter sets. A parameter set employs structure and configurable options as follows.

- **Parameter to monitor:** This option contains a pointer to the information to be captured. Multiple instances of the same parameter may be created to allow using it with different options.
- **Data retrieval:** Data is requested or challenged in regular intervals, on request or in a push model. Based on this, a check on certain thresholds or pass/fail tests is performed. In interval mode, an interval time which may be multiples of the global interval time is set. If the interval time already exceeded when the set is retrieved for data, a new parameter request is performed. Parameters obtained in interval mode on regular interval basis are valid for one whole interval time.
A data retrieval operation on request by another instance, i.e. another metric set, is another configurable option to allow metric sets to retrieve actual data.
A third option is the usage of push operations which need no monitoring based on regular intervals, as an action is requested by an external module, i.e. based on a parameter change monitored by an external module. Triggered operation may be enabled as push operations to define if a certain parameter causes SFM to perform failure operations immediately. These parameters may be binary, like a case-open detection, where push operation is initiated by an external mechanism. In this way, not the data itself is requested but the trigger only which assumes some kind of context awareness. Push operated parameter sets are marked as push executable only.
- **Threshold/Action:** Multiple threshold and according emergency profiles are stored in the node. Therefore, different actions may be performed on certain threshold levels. This includes different services to deregister or to exploit other local energy saving mechanisms, like disabling of certain peripheral components. Also, different requests to FABD and/or RB modules may be defined here.
If the set only serves as parameter to be requested by a metric set, no thresholds and actions may be defined here.

A metric set only has 4 configurable options. A metric is calculated based on one or more parameters or other metrics.

- **Metric parameters:** The metric that has to be calculated may consist of several parameters or other metrics resp. their memory, so a list of other sets is added here. If the metric includes a parameters which data is retrieved by the push mechanism, the whole metric is marked as push executable only.
- **Metric calculation:** The formula for calculation based on the queried sets and memory is defined.
- **Metric Memory:** A metric may have several memory slots for storing its old values or other values needed for calculation in case this is needed for operation. For example, changes of certain parameters can be detected.
- **Threshold/Action:** Multiple threshold and according emergency profiles are stored in the node. Therefore, different actions may be performed on certain threshold levels. This includes different services to deregister or to exploit other local energy saving mechanisms, like disabling of certain peripheral components. Also, different requests to FABD and/or RB modules are defined here.

In case a metric has to be calculated, no parameter querying is done but a parameter set has to be defined separately which is requested to offer the information. Also, the parameter set defines if the parameter has to be explicitly requested or if the last retrieved value from last interval scan is taken based on the option set in Data retrieval. Push operations are not supported by metrics.

A Threshold/Action item of a Threshold/Action list list contains the following items:

- **Maximum value:** The upper bound on the parameter or metric to be checked.
- **Minimum value:** The lower bound on the parameter or metric to be checked.
- **Enabled:** The according item may be disabled or enabled on request.
- **Repeat flag:** An action may be applied only once when entering the bounds or applied repeatedly when staying within the bounds for several calls of the Threshold/Action item.
- **Action:** A pointer to the action to be taken when the Threshold/Action item is valid.

The according protocol regularly loops over all sets based on a global interval time and takes the according actions. First, metric sets are processed as they already may include parameter sets which scan in certain intervals. Therefore, the situation can be avoided that a parameter set is executed and new data is retrieved which is intended to be used by a metric set but is obsolete when the metric set is processed.

In case of metric sets, parameter sets are queried and the metric is calculated. Based on the thresholds, actions are performed. Push executable metric sets are ignored here.

In case of parameter sets, the data is retrieved according to the Data retrieval options, thresholds are compared and the according actions are performed. Push executable parameter sets are ignored here.

Several endpoints may be defined that serve as input for push operated parameter sets. If SFM receives a pushed request, the according metric and parameter sets are processed immediately.

In case a metric includes a pushed parameter, all parameter sets of the metric are processed now and according actions are performed.

Pushed requests are sent by external modules which need immediate awareness or where an external mechanism is installed. For example, if a node is able to detect if its case is opened, a pushed request could be sent to shut down the node. This allows the node not to take care of regularly monitoring certain parameters and thereby save some energy.

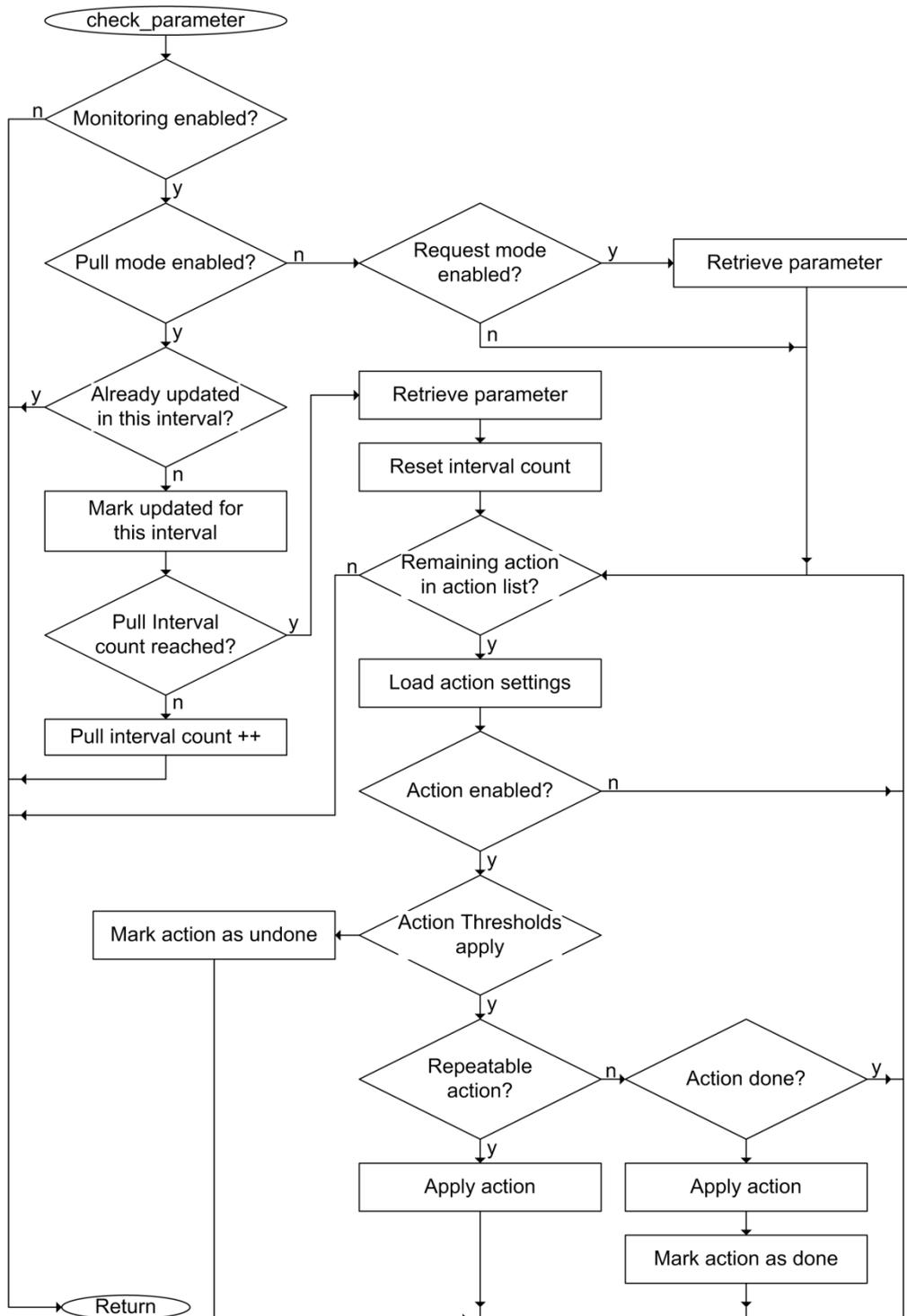


Figure 2: Flowchart of a single parameter check; first, the mode and interval are checked, then thresholds and actions are applied

In Figure 2, the protocol for checking of a single parameter is presented. This flow is applied each time a single parameter needs to be checked in certain intervals (pull mode), on request by another metric (request mode) or by an external module (push mode). After checking the mode, the parameter is updated and according thresholds are checked. In case the parameters fulfill the checked threshold limits, associated actions are applied. Actions may be repeated in any interval as long as the parameter value fulfills the threshold limits or applied only once when entering the limits.

Based on this structure, an easily extendable self-monitoring mechanism is established. For the first trial, a simple energy metric is defined which reflects a self-evaluation of the nodes energetic state. For battery powered devices, the current battery state is divided by the full battery state which gives a percentage of remaining energy. If this falls below certain pre-defined threshold, local energy saving possibilities are exploited by activating the according emergency profile. This includes minimizing the transmission power and extending wake-up duty cycles. If available, additional services are deregistered, i.e. demanding security capabilities or collaborative monitoring actions, keeping only the main sensing or acting functionality of the node running. FABD is contacted to disseminate node failure that further actions can be taken by the application, i.e. requesting battery maintenance.

4.2 Failure Anticipation

4.2.1 Tasks and Actions

While most existing resilience solutions address error detection and reaction, failure anticipation aims at preventing failures or mitigating their criticality *prior to their occurrence*. This functionality is offered in TWISNet by the Failure Anticipation module (FA).

Albeit it is specified in what follows, the Failure Anticipation module will not be implemented in TWISNet, given the heavier emphasis put on reactive solutions involving resilience, threat detection and security adaptation.

As opposed to the Sensor Failure Management module, the Failure Anticipation module requires exchange among nodes. While the Sensor Failure Management may locally induce that a failure is about to happen (e.g., by monitoring that a node is running low on battery), the Failure Anticipation module targets more complex failure which occurrence is dependent on interactions between nodes, and which detection is therefore requiring the involvement of a plurality of nodes.

Two purposes are foreseen: determination of the incurring delay for a process involving more than one node, and identification of a criticality metric for each node. The former considers that multiple nodes are typically involved in the fulfilment of a service (data plane or control plane), and that a mechanism should be provided in order to globally determine global latency for a given service and to determine whether this latency is acceptable or may lead to a possible failure. The latter takes into account the security service(s) offered by the node, the available equivalent/fallback solutions in the vicinity and the interdependencies of nodes between each other. From these parameters, it determines a criticality value assigned to a node, which is representative of the node's importance within the network and the consequences of its possible failure on its environment.

The underlying technical solutions enabling failure anticipation are based on the determination of global parameters expressing either delay (in case of data and/or control plane packets) or nodes interdependencies while considering the fulfilment of a global objective, such as the delivery of a data packet or the operation of a security service. Another key concept that is introduced in order to determine whether a faulty node would lead to an unacceptable service interruption is the concept of exercise, wherein a non-faulty node

simulates global failure, so as to let its peer(s) attempt to rely on fallback solutions. Specific messaging has to be introduced accordingly to make sure that a critical situation will not arise from an exercise attempt.

The return values of the Failure Anticipation module may either be used by the Self Healing module, or may be used to raise a flag to the administrator, signifying that a failure is likely to happen or that a certain failure incurring would lead to unacceptable consequences.

4.2.2 Protocol

Failure anticipation as it is designed in TWISNet is not provided on an applicative basis, but rather relies on lower-layer mechanisms, the user being eventually notified when his intervention is required, or as part of regular logging operations. This is motivated by the fact that the resource-constrained sensors will likely not be able to accommodate the management of complex applicative operations, especially on a regular (surveillance) basis. Furthermore, limited battery power would not advocate the use of dedicated send/receive operations for optional functionalities like failure anticipation

Rather, the preferred transport approach for failure anticipation-related traffic is to be piggybacked along with other messages that the nodes forming the constrained topology will exchange for applicative or control purpose.

4.2.2.1 Delay monitoring

The delay monitoring operation requires that a node forwarding a packet adds a δt information expressing the forwarding delay for this packet. δt is computed as the difference between the reception time (or issuance time if the node is the sender) and its delivery time. If a reliable delivery mechanism is used between the node and the neighbor to which the packet is forwarded, information from this delivery mechanism should be taken into account so that, the last (and successful) delivery attempt is the one that is used for the delay computation. Information relevant to the global delay for a single packet delivery (from source to recipient) is Δt , which is obtained as the sum of individual delays δt . At each node i , Δt is increased by δt_i and the local value δt_i replaces the one that was present in the received packet, as depicted in Figure 3.

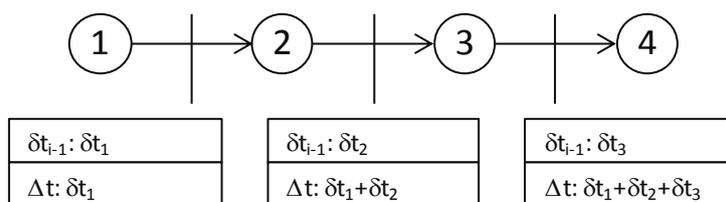


Figure 3: In-packet delay information.

Failure anticipation for delay monitoring consists for the final recipient of the packet to check whether the evolution of global delay Δt is compatible with its own requirements. Simple mechanisms should be used to derive the trend in overall latency evolution and ensure that the packet delivery will remain below an acceptable threshold. The same kind of operation can be performed at each intermediary node, by checking whether the received δt_{i-1} values for a same neighbour are not evolving towards an unacceptably high value. In either case, a warning should be issued to the administrator.

For 6LoWPAN networks, the best identified place to carry delay monitoring information is within the IPv6 header, especially, within an IPv6 hop-by-hop option extension header.

4.2.2.2 Interdependency status & criticality metric

Assessing nodes' interdependency in the fulfilment of a service is a complex task in multiple regards. First, interdependency itself must be properly qualified: for example, two nodes may be required to be authenticated prior to interacting with each other. Depending on how interdependency is meant, this may lead to make to include authentication components in the interdependency nodes set for these two nodes and the service they are to enable. Second, a place must be determined at which interdependency can be observed and translated into a criticality metric. Finally, a protocol must be specified that corresponds both to the initial set of requirements (transport information on nodes interdependency) and to the capabilities of the nodes that run it.

A first option could be for a node to include information about the peers on which it is dependant for the fulfilment of a given service. Then a third party entity would monitor these pieces of information and induce global metric values for all nodes. However this option is not realistic, since a client node is likely not to have any knowledge on the infrastructure nodes that participate in the provision of a service to it. Furthermore, it would quickly lead to a communication overhead that would not be acceptable in a wireless sensor network. A second, preferred, option consists for each node to maintain by itself a metric representative of the number of its peers with respect to a given service.

From a common initialisation time, we propose that nodes offering a given service (for example, PANA Authentication Agent service) increase a value corresponding to this service when they interact with a peer. Note that this requires that the serving node be aware of the clients' privacy scheme in order to distinguish a really new peer from one having just acquired a new pseudonym. In case of interaction with an already known peer (e.g. performing a re-authentication for a node), the increment value should be lower. The increment value should be increased when a simulated failure (exercise) has shown that a client does not have a fallback solution.

At current stage, no detailed protocol specification is provided regarding the way for the human operator to access the criticality metrics. A basic client/server protocol (such as CoAP) with an adequate security scheme is seen as a good candidate.

4.2.2.3 Failure exercise

The failure exercise scenario is used as a means by a node to let its clients (the nodes to which it is offering a security service) determine whether they have a fallback solution in case of its unavailability.

Upon an access attempt to a service offered by a server node, this latter should answer with a specific message indicating its (pseudo-) failure, declared for exercise purpose. This message includes information on how to override the exercise behaviour of the server node, e.g. giving a codeword to be included by the client node in a subsequent request, if this latter was not able to identify a fallback solution. In this case, the criticality metric for the server node is increased by a higher increment.

4.3 Failure and Abnormal Behaviour Detection

4.3.1 Tasks and Actions

The Failure and Abnormal Behaviour Detection is composed of two modules: one on the node and one on the Application Server.

The FABD on the sensor side is responsible for detecting current and future failures of the node by sending data stored locally to be analysed on the FABD server side. It also receives the information provided from the FABD on the Application Server regarding the state of

health of the node itself and passes the information on to the Sensor Failure Management module to be processed.

Additionally, the FABD on the node side may receive indications from the FA or from the SFM module that something is wrong. If there are problems connecting to certain neighbours a warning may be sent to the FA and one to the FABD server to investigate if the problem is external. Consequently, if the problem is local, an appropriate message is sent to the SFM that will treat it. If a message from the SFM module is received suggesting that something is wrong and whether it is treatable or not, a message is sent to the FABD server announcing that there are difficulties. The sensor may receive feedback or commands from the server FABD and, if the failure is critical, it can receive a shutdown message. Communication and data problems are handled in a similar manner to the connection issue. The FABD server that can send back one of the following messages: it may reassure the sensor that the problem is not local (case in which it may recommend to the Secure Routing module to avoid communication with the failing neighbour node or nodes), deliver hints for solving the problem or tell the node to shut down if it has failed and cannot recover.

The FABD module on the Application Server stores information that can differentiate between a healthy node and one that has been corrupted or sends incorrect data. It communicates with the FABD module on the sensor by receiving sample messages from it and classifying them into one of the categories stated previously. It can proceed to send this result to the inquiring node and may even send optional advices on how the problem should be treated. Moreover, the FABD module on the Application Server can periodically contact the nodes for evaluation purposes.

The FABD on the server side alerts a node if it is in an undesired state. Because it processes information coming from multiple nodes, it can find out which node is causing the problem and act accordingly. If this is a major problem, an administrator will be alerted. If a node is in a state in which other nodes should not communicate with it or other nodes should be able to choose whether they should communicate with it given its state, the FABD Application Server will send information about the state of the node to the Secure Routing module. The Secure Routing module manages this information and decides whether communication with a certain node should be established or not.

Monitoring a group of neighbouring nodes is reduced to data problems, but this functionality can also be extended to handle connectivity and communication problems as well. The FABD Application Server compares the messages from each neighbouring node. If a node is singled out, an appropriate message will be sent to the respective FABD node side to investigate the problem and see if that node is failing. Communication problems may be detected if there is a percent of lost packets over a certain threshold, whereas data problems can be detected as spikes (big differences) between what we expect from that node, what are our results and what are the results of other neighbours.

The FABD on the server side receives messages from nodes concerning themselves or their neighbours. A sensor will send messages to the server FABD to check if a certain behaviour implies the possibility of a failure (as a request from its FA or SFM modules) and it will receive feedback and an optional suggested action (preventive, respectively corrective or recovery messages). A sensor may have suspicions regarding a node or it may be certain that the node has failed. Consequently, the node will send a message to the FABD server with its observations regarding the suspected neighbour. If the FABD server has more complaints about the neighbour sensor, it will be marked as corrupt. However, if the complaining node is suspected to be corrupt and its neighbour has had no previous problems, the node itself will be the one marked. In these two cases the marked node will be treated as explained in the next paragraph. There could be cases in which both nodes

(complainer and neighbour) are equally healthy or problematic. This will result in marking both nodes and sending warning messages about them in the network.

The Application Server FABD will broadcast to all the Secure Routing modules messages revealing that a particular node has been compromised and the healthy nodes should stop communication with it. This helps consolidate the trustworthiness of the network by minimizing the replication and processing of invalid data coming from problematic nodes. Moreover, the energy consumption is reduced because the communication with the failed sensors is stopped. This decision is taken after carefully analysing the messages received from the presumably corrupt sensor and the messages received from its neighbours over a period of time. After marking a node as being corrupt, the FABD server side sends a message to the node in question suggesting it should investigate the problem. It might even give hints about what the problem is: communication, bad data etc. It then receives a message from the sensor FABD stating that it has healed, no problems were found or that the failure is critical. It can choose to rehabilitate the node if one of the first two cases appear, but the node will be put in a “trial” period, meaning that all the SR modules receive a warning concerning that node and they may choose not to communicate with it if the information they require is critical. The warning has a timeout on the local sensors, if everything goes well no additional data has to be sent in order for the node to be completely reinstated. However, if problems still arise the Application Server FABD may choose to shut down the node: sends a message to the failed sensor indicating it should shut down because it is compromised and broadcasts a message to all SR modules to blacklist that sensor (in case it doesn't follow the shutdown command).

The FABD server can also manage statistical data in order to predict node failures. For instance, the server can keep track of node uptime, the number of messages sent/received by a node etc. and notify an administrator when certain thresholds are met.

4.3.2 Protocol

The data analysis process takes place only on the FABD server because it has a wider perspective about the network and the data that flows through it. The FABD server can detect spikes that take place occasionally and monitor their frequency. Detection can be done by inspecting a number of sequential packages and comparing them one to another, using a threshold. If a value is too big or too low compared to the others and if it is an isolated case, it is considered to be a local exception. Moreover, data samples from the sensors will be used to realize a benchmark for the nodes and generate trust values for them. Given the fact that the data provided by the nodes depends on their localization; it is unfeasible to provide the FABD server with initial assumptions about the packages. We may, however, provide the FABD server with an initial grouping of the nodes and compare the packages of nodes only relative to the nodes in the same group. Considering this approach, we could consider an initial probation period in which every node is considered to have healthy data and, through comparison as stated before, we would adjust our beliefs about the healthiness of the sensors.

A node can find itself in one of the following states: healthy, failing or failed. The failure protocols refer to these sensor states and model the situations as follows. On the server side, these states are further expanded into: healthy, backuping / restoring, exercise fail, failing, failed, critical, recovering. The transitions between these states can be seen in Figure 4.

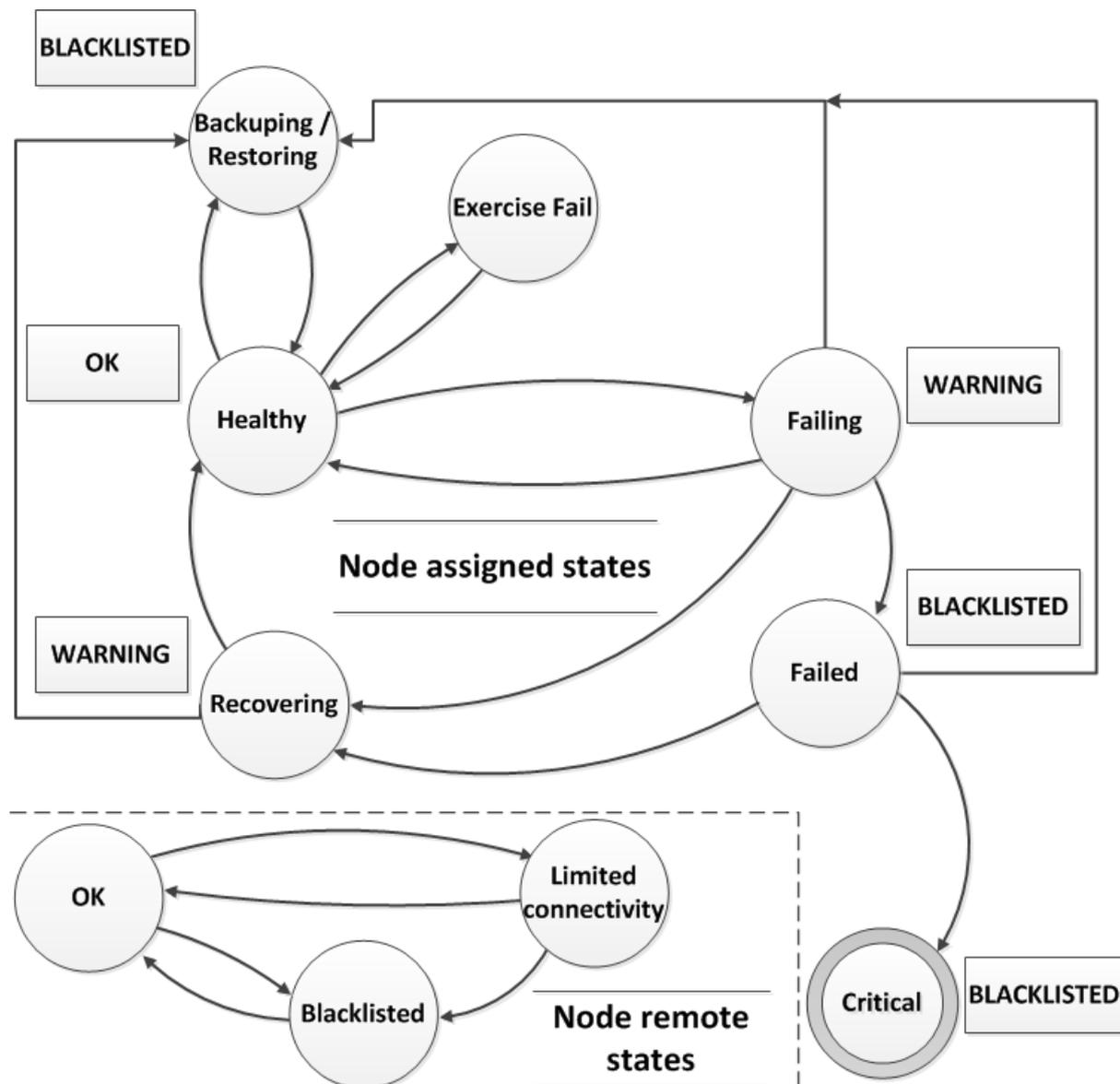


Figure 4: The states that the FABD node knows of and the states the FABD application server is aware of.

If the node is healthy no actions are taken against it. It will continue to work according to its program, but occasionally the FABD server may ask it for its status (data samples, optional connection information etc.). This data is inspected and the nodes will be categorized as stated before and appropriate actions will be taken. The data is furthermore processed to help monitor the other nodes in the network and place them in the corresponding category.

If a node has failed, it has to be decided if it can recover from its failure by itself or if it should be shut down in order to avoid compromising the entire network. If it can recover it must be first marked as inactive to the other nodes (through a broadcast message to the SR module) and wait for a message of healing completion from the node. The healing status of the node will be provided to the FABD node side by the SFM module and it will be passed on to the FABD server. The node might also respond that it could not heal, case in which it shuts down (without awaiting a message from the FABD server to avoid bandwidth usage and energy consumption); the FABD server sends a message to all SR modules to mark this node as terminated (to make sure that it doesn't keep sending messages in the network). If there was

a positive response and the sensor considers that its healing is complete, the FABD server will broadcast a message to all SR modules marking it as active, but with a warning attached that the node is still suspected until a timeout expires. Some nodes may chose not to communicate with this sensor, but the FABD server does not impose this decision. If the timeout has expired and none of the nodes that communicated with this sensor have complained about it then the FABD server does nothing because the timeout is local to every sensor after receiving the warning.

A node is considered to be failing because it sometimes has problems such as data spikes, but this may be extended to consider connection or communication problems as well. These problems should be treated internally by the sensor, but there may be cases when the sensor cannot realize that it has a local problem. The FABD server, however, gathers information from all nodes and it can identify if a node is faulty. In this case, it sends a message to the FABD part of the sensor and it may also give hints towards the nature of the problem. If this is a recurrent problem, the FABD server may choose to inform the other sensors that this node is faulty by broadcasting a warning message to the SR module. This type of warning message does not expire on the sensor nodes. However, the FABD server will keep collecting information regarding that node and when (and if) it considers the node healed it will send a message to clear the warning sent before.

A basic flowchart of actions can be observed in Figure 5.

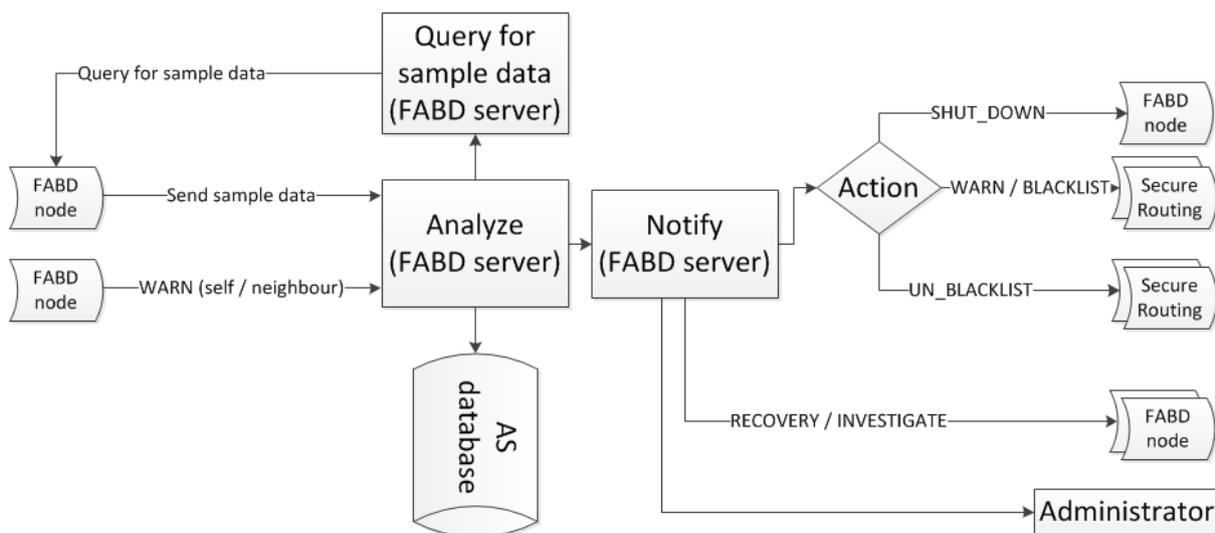


Figure 5: Flowchart for the FABD actions.

4.4 Resilience Backup

4.4.1 Tasks and Actions

The Resilience Backup module is responsible for management of schedules for creation and transfer of partial configuration or full backups of the node’s configuration values to the Resilience Database and for recovery from backups in case of failure. It uses functionality of the SMDR module to request actual parameters and for insertion of recovered parameters. In case of management of firmware backups, an optional interface with the SMDU is needed.

Backups of configurable parameters are scheduled regularly to have a defined fall-back state in case of recovery. Also, other layers may request backups at a dedicated time, i.e. before they perform a critical operation, to guarantee a smooth recovery if the critical operation fails.

A service for recovery is offered which may be used by context aware modules which detect failures or risks. If for example Threat Detection signals a failure and decide that a complete or partial recovery is needed, a request may be sent to RB which takes the according actions.

Backup functionality is needed in scenarios where availability is required and manual maintenance is difficult. Especially, in industrial long-term installations, automatic parameter backup and recovery is useful. In case of battery-powered nodes, recovery may be requested after battery maintenance was performed.

4.4.2 Protocol

For backup, two cases can be separated. Regular backup is handled through schedules stored on the RB server module installed in the mediation layer. A server schedule entry contains

- a start time which serves as reference for backup operations,
- an optional end time in case backup is needed only over a certain time span,
- a time interval representing the time interval when a new backup has to be taken,
- the node address or node group from which the backup has to be taken,
- the parameters that have to be backup-stored which could also be the whole firmware,
- an update flag that presents if a new backup of the same type replaces the old backup,
- a copy flag, indicating if a copy of data to a dedicated RB database is needed or if the data only remains in the SMDR database from where it can be read for recovery
- and a storage time that represents the time that a certain backup has to be kept on the server in case a new backup does not replace the former backup.

The RB module is responsible for applying backups at the configured times using the SMDR parameter database as well as the SMDU database. Parameters are read from this database and logically copied to a RB database ideally placed on a physically separated server.

By using the update flag it is possible to keep different configurations of a node on the resilience server which may serve as fall-back states.

For a firmware backup, it is convenient to backup the firmware version number only, represented by a hash or an ID, that a firmware recovery may be installed from a firmware source where the original firmware is stored. Additional configuration parameters are backup-stored and recovered separately.

The same applies for configuration values in case the SMDR database is configured to hold historical values which may be used for recovery. In this case, the copy flag can be unset to allow a logical copy only, lowering resilience and availability of the data but saving traffic and storage in the RB database.

The backup parameters include the source and the parameters as well as a timestamp of the backup. Backups are replaced or deleted according to the update flag and the storage time.

Also dedicated requested backups are possible where no schedule is used. The requested backup parameters include the same information like a schedule entry except for start, interval and end time. It additionally includes an ID to identify the backup data set.

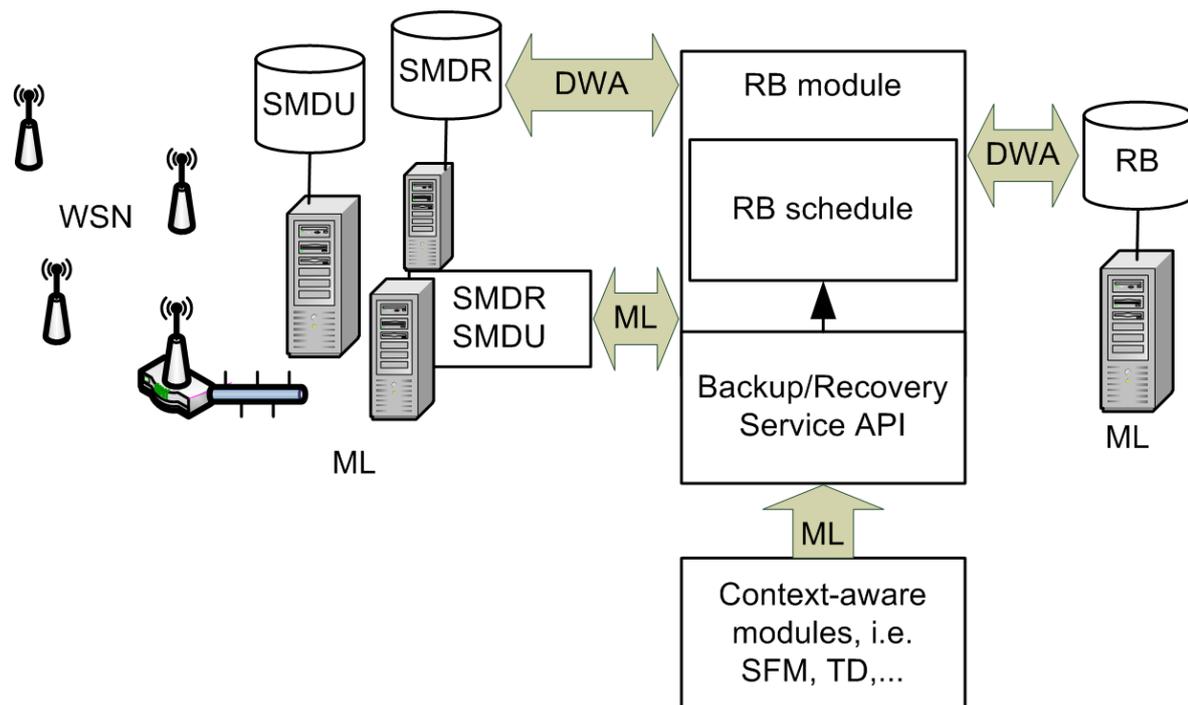


Figure 6: Intercommunication of RB with other modules; SMDR/SMDU values are backup-stored to RB database according to schedule; recovery and backup requests may be queried by ML calls to the API

For a recovery request, a context-aware module sends a request for recovery to the RB server module. The RB module reads the data from its recovery database and writes the according parameters into the database of SMDR. Afterwards, it queries SMDR to perform a write operation to a certain node. In case data was not actually copied to the RB database but historical data is stored in the SMDR database, RB reads the according values from this database to proceed.

A recovery request includes:

- the destination node address that shall be recovered,
- the source node address of the backup,
- a start time representing the earliest time that a backup may have to apply for this recovery,
- an end time representing the latest time that a backup may have to apply for this recovery,
- an ID if a recovery from a requested backup is requested and
- the parameters to recover and
- the order they have to be recovered.

The separation of destination and source address is useful to recover a parameter set of a certain node to another node.

When a request for recovery arrives, the backup server checks when the last backup of the requested parameter has been done that satisfy the requested constraints. In case several parameters have to be recovered, the order of recovery has to be considered to avoid undefined device states.

During recovery, the RB module notifies FABD that the device is in recovery mode, where no other requests are processed. This protects the device from running into undefined states or answering on requests with insufficient security configuration applied.

A recovery after maintenance due to damage and exchange of a node could include recovery of firmware first by using the firmware version number only and requesting the firmware from a central server and applying the backup configuration of the replaced node afterwards.

4.5 Self Healing

4.5.1 Tasks and Actions

Local self-healing action involves collaboration among nodes to restore a service that become unavailable due to the failure of one or more entities. Collaborative actions can be of two types: punctual collaborative actions that only occur during the recovery process, or long lasting actions in which a set of nodes collaborate to jointly enable an interrupted service, on a distributed basis.

The module considered in this section mainly focuses on punctual collaborative actions for the recovery of a security service. This process assumes a three phase operation: 1) nodes discover locally the failure of a security service; 2) nodes establish among themselves the best recovery solution; 3) nodes apply the agreed recovery solution.

4.5.2 Protocol

The two following kinds of protocols exchanges are required by this module:

- **Exchange of information about nodes' capabilities and running security services.** This is required in order to discover the failure or reappearance of a local security service, as well as to discover whether a node, without currently running a security service, could possibly enable it if needed.
The set of supported security services include:
 - Ability to act as a PANA Enforcement Point for domain D;
 - Ability to act as a PANA Relay for domain D;
 - Ability to act as a PANA Authentication Agent for domain D.

The node provides information about the security services it is running or may support using predefined syntax, in the form of a dictionary of (*service_id*, *service_status*) bundles wherein the *service_id* are statically known to the nodes that make up the topology. Use of such static identifiers (instead of, for example, XML-based syntax) aims at saving resources for the constrained devices. Likewise, advertisement of supported and running security services should not involve dedicated protocol exchanges, but rather be piggybacked on top of otherwise mandatory exchanges between constrained nodes (such as routing messages) in order not to spare nodes energy.

- **Exchange of nodes' configuration information, when putting in place a fallback solution.** The actual mechanism for re-enabling a faulty service is still under development. Two distinct approaches are explored: either the self-healing process happens on a purely local basis, or a higher-level entity (such as gateway, backup server

or AAA server) plays a major role in it. The first approach is traditionally itself split into two schemes, depending on whether an *election* happens for designated the node that will replace the faulty one, or the service will be offered on a distributive basis. In the second approach, the higher-level entity designates the node that will replace the faulty one.

In most security scenarios, a service is offered on a hierarchical basis. For example, an enforcement point, which establishes a secure tunnel with a newly authenticated node for the purpose of enforcing network access control, refers to the network access server through which authentication was performed. Or the network access server is itself the client of a AAA server. The scenarios pertaining to these services therefore eventually require that the entity to which the replacement node will report takes part to its designation. Therefore, the corresponding services must be re-enabled upon failure by decision of this higher-level entity, which excludes the first, local approach described here above.

While piggybacking of information over the routing protocol exchanges is the preferred option with respect to discovery, more complex information exchanges during re-configuration phase require higher bandwidth and responsiveness, in order to restore service availability as quickly and efficiently as possible. The use of a lightweight, secure applicative protocol such as the CoAP protocol is therefore the preferred alternative.

The Self Healing module is an optional module of the TWISNet framework. It will not be implemented, given the heavier emphasis put on reactive solutions involving resilience, threat detection and security adaptation.

5. CONCLUSION

This deliverable described the security solutions related to task 3.4 that are going to be exploited in the TWISNet security framework.

Starting with a state-of-the-art study, the security solutions focused on providing availability on information and communication are described in scope of trustworthy assessment and resilience solutions.

The deliverable presents the architecture of the task framework with the identified security solutions as modules. The architecture description shows that the identified modules interact mostly with each other. They may interact with other modules from the TWISNet framework like for instance the key management module or the secure update mechanisms. A detailed description of these modules is presented.

The Sensor Failure Management module monitors the sensor device locally and uses algorithms to detect failures that take place at the device level. In case of failure, recovery methods are employed to eliminate the failure and its effects.

The failure anticipation module works on a peer-to-peer basis by interacting with similar modules on other sensor nodes. The objectives of this module are to evaluate the local risk of failure of a node and to perform collaborative actions with its peers in order to achieve global failure anticipation. Global failure anticipation is also intended to prevent domino effects: by evaluating their respective dependencies and the available fallback solutions, nodes should be able to anticipate chain reactions and to identify single point of failures. The Failure Anticipation module is an optional module of the TWISNet framework.

The Failure and Abnormal Behaviour Detection module monitors the sensor devices for abnormal output with server assistance in order to detect node failure or the failure of neighbours.

The Resilience Backup module is responsible for managing schedules for creation and transfer of the partial configuration or full backups of the node's system to a Resilience Server and for recovery from backups in case of failure. Backup is done periodically or on request.

The self-healing module in a sensor node collaborates with other similar modules within other nodes in order to enable a collaborative resilience system between nodes. Through this module, nodes exchange information relative to their needs and capabilities in terms of security. This is especially relevant when, because of some unanticipated failure, a security function offered by a node A to a node B stops being available. The Self Healing module is an optional module of the TWISNet framework.

The different described modules fit the general picture of the trustworthiness and availability framework.

6. ACKNOWLEDGEMENT

The TWISNet consortium would like to acknowledge the support of the European Commission partly funding the TWISNet project under Grant Agreement FP7-ICT-STREP-258280.

7. REFERENCES

- [1] N. Oualha and Y. Roudier, "Securing Ad Hoc Storage through Probabilistic Cooperation Assessment", *Electronic Notes in Theoretical Computer Science (ENTCS)*, v.192 n.2, p.17-29, May, 2008
- [2] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and secure sensor data storage with dynamic integrity assurance", In *Proceedings of IEEE INFOCOM 2009*, Rio de Janeiro, Brazil (April 2009)
- [3] T. Park and K. G. Shin, "Soft tamper-proofing via program integrity verification in wireless sensor networks", *IEEE Transactions on Mobile Computing*, 2005
- [4] Chen, D. & Varshney, P. K., "QoS Support in Wireless Sensor Networks: A Survey". *International Conference on Wireless Networks*, 2004
- [5] M. Aykut Yigitel, Ozlem Durmaz Incel, Cem Ersoy. "QoS-aware MAC protocols for wireless sensor networks: A survey". *Computer Networks*. 2011
- [6] Jiang, S., "Network and Service Failure Restoration and Prevention in Multi-Hop Wireless and Mobile Networks", PhD Thesis, December 2009
- [7] Yuan Guo; McNair, J., "Redundancy versus lifetime tradeoff analysis for environment monitoring using wireless sensor networks," *Dependability and Security in Sensor Networks and Systems, 2006. DSSNS 2006. Second IEEE Workshop on* , vol., no., pp.7 pp.-77, 24-28 April 2006
- [8] Tezcan, N.; Cayirci, E. & Caglayan, M. U., "End-to-end reliable event transfer in wireless sensor networks.". *PIMRC'04*. 2004
- [9] Ahn, J.-S.; Hong, S.-W. & Heidemann, J., "An Adaptive FEC Code Control Algorithm for Mobile Wireless Sensor Networks". *Journal of Communications and Networks*. 2005
- [10] Zhang Liankuan, Xiao Deqin, Tang Yi, and Zhang Yang, "Adaptive error control in wireless sensor networks". *IET International Conference on Wireless Sensor Network*. 2010
- [11] Moshaddique Al Ameen, S.M. Riazul Islam, and Kyungsup Kwak, "Energy Saving Mechanisms for MAC Protocols in Wireless Sensor Networks". *International Journal of Distributed Sensor Networks*. 2010
- [12] Demirkol, I. and Ersoy, C. and Alagoz, F., "MAC protocols for wireless sensor networks: a survey". *Communications Magazine*, IEEE. 2006
- [13] Abbas, Ash Mohammad and Kure, Oivind, "Quality of Service in mobile ad hoc networks: a survey". *Int. J. Ad Hoc Ubiquitous Comput*. 2010
- [14] Li, Yanjun; Chen, Chung Shue; Song, Ye-Qiong and Wang, Zhi, "Real-time QoS support in wireless sensor networks: a survey". *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems*. 2007
- [15] Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), *IEEE Standard 802.15.4-2009*
- [16] L. Paradis et. al., A Survey of Fault Management in Wireless Sensor Networks, *Journal of Network and Systems Management*, Springer Netherlands, Issue Volume 15, Number 2 , June, 2007, Pages 171-190

- [17] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. K. Khosla, "SCUBA: Secure Code Update By Attestation in sensor networks", ACM Workshop on Wireless Security, WiSe 2006, ACM, 2006, pp. 85-94
- [18] F. L. Lewis. "Wireless Sensor Networks", in Smart Environments: Technologies, Protocols, and Applications ed. D.J. Cook and S.K. Das, John Wiley, New York, 2004
- [19] Joaquin Canada-Bago, Jose Angel Fernandez-Prieto, Manuel Angel Gadeo-Martos and Juan Ramón Velasco "A New Collaborative Knowledge-Based Approach for Wireless Sensor Networks" <http://www.mdpi.com/1424-8220/10/6/6044/pdf> [Accessed: May 2011]
- [20] FRBS - Fuzzy Rule-Based System http://www.perlmonks.org/?node_id=577755 [Accessed: May 2011]
- [21] G. Gupta and M. Younis, "Fault-tolerant clustering of wireless sensor networks", Wireless Communications and Networking, 2003, IEEE
- [22] C. Wan, S. Eisenman, and A. Campbell. 2003. CODA: congestion detection and avoidance in sensor networks. In Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys '03)
- [23] Y. Qian, L. Lu and D. Tipper, "A design for secure and survivable wireless sensor networks", IEEE Wireless Communications, 2001.
- [24] J. Saia and A. Trehan. "Picking up the pieces: Self-healing in reconfigurable networks". In IEEE International Parallel & Distributed Processing Symposium, 2008.
- [25] T. Hayes, N. Rustagi, J. Saia, and A. Trehan. "The forgiving tree: A self-healing distributed data structure". <http://www.cs.unm.edu/amitabh/pubs/ForgivingTreeNV19-525P.pdf> [Accessed: May 2011]
- [26] M. Lanthaler. "Self-Healing Wireless Sensor Networks", Seminar on Self-Healing Systems, Department of Computer Science, University of Helsinki, Spring 2007.
- [27] A. Trehan, J. Saia. "Scalable and Distributed Self-Healing Algorithms for Reconfigurable Networks", in IEEE Computer Applications in Power, 2001.
- [28] T. Bokareva, N. Bulusu, S. Jha. "SASHA: Toward a Self-Healing Hybrid Sensor Network Architecture", in Proc. of The Second IEEE Int'l Workshop on Embedded Networked Sensors, May 2005.
- [29] N. Sastry and D. Wagner, "Security considerations for IEEE 802.15.4 networks," Proceedings of the 3rd ACM workshop on Wireless security, New York, New York, USA: ACM, 2004, pp. 32-42
- [30] D. Raymond and S. Midkiff, "Denial-of-service in wireless sensor networks: Attacks and defenses," IEEE Pervasive Computing, vol. 7, 2008, p. 74–81
- [31] D. Raymond, R. Marchany, M. Brownfield, and S. Midkiff, "Effects of Denial of Sleep Attacks on Wireless Sensor Network MAC Protocols," 2006 IEEE Information Assurance Workshop, 2006, pp. 297-304
- [32] L. Sa de Souza, "FT-CoWiseNets: A fault tolerance framework for wireless sensor networks," International Conference on Sensor Technologies and Applications, SensorComm 2007, IEEE, 2007, pp. 289-294
- [33] H. Ishikawa, A. Courbot, and T. Nakajima, "A framework for self-healing device drivers," Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, IEEE, 2008, p. 277–286
- [34] B. Li, R. Doss, L. Batten, and W. Schott, "Fast recovery from node compromise in wireless sensor networks," 2009 3rd International Conference on New Technologies, Mobility and Security (NTMS), IEEE, 2009, p. 1–6

- [35] C. Ma, X. Lin, H. Lv, and H. Wang, "ABSR: An Agent Based Self-Recovery Model for Wireless Sensor Network," 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, IEEE, 2009, pp. 400-404
- [36] <http://monet.tekever.com/> [Accessed: May 2011]
- [37] <http://www.eu-mesh.eu/> [Accessed: May 2011]
- [38] <http://www.netqos.eu> [Accessed: May 2011]
- [39] <http://www.aroma-ist.upc.edu/> [Accessed: May 2011]
- [40] <http://www.ana-project.org/> [Accessed: May 2011]
- [41] SENSEI Project : <http://www.sensei-project.eu> [Accessed: May 2011]
- [42] WSan4CIP Project : <http://www.wsan4cip.eu> [Accessed: May 2011]
- [43] TAMPRES. Available: <http://www.tampres.eu/> [Accessed: May 2011]
- [44] INSPIRE. Available: <http://www.inspire-strep.eu/> [Accessed: May 2011]
- [45] INTERSECTION. Available: <http://www.intersection-project.eu/> [Accessed: May 2011]
- [46] AMBER. Available: <http://www.amber-project.eu> [Accessed: May 2011]
- [47] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994
- [48] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, Dec. 1998
- [49] Institute of Electrical and Electronics Engineers, "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements)", IEEE Standard 802.11e-2005, 2005
- [50] HomePNA Alliance, "HomePNA v3.1: 320 Mbps Home Networking Specification", Dec. 2006
- [51] D. D. Clark, "Fault Isolation and Recovery," RFC 816, July 1982
- [52] Evaluation Engineering article on commercial WSN solutions <http://www.evaluationengineering.com/index.php/solutions/instrumentation/wireless-sensor-networks-are-taking-over.html> [Accessed: May 2011]
- [53] Dust Networks technology <http://www.dustnetworks.com/technology> [Accessed: May 2011]
- [54] Millennial Net nodes using the D3R algorithm for repairing and frequency hopping <http://www.millennialnet.com/technology/m5advantages.php> [Accessed: May 2011]
- [55] Green Peak chips with two separate antennas <http://www.greenpeak.com/Product/Chips.html> [Accessed: May 2011]
- [56] Protecting Mission-Critical Workloads with VMware Fault Tolerance, VMWare White Paper, 2009. http://www.vmware.com/files/pdf/resources/ft_virtualization_wp.pdf [Accessed: May 2011]
- [57] DFT / HA Reliability. Available: <http://www.ccpu.com/trillium-protocol-software-products/dft-ha-distributed-fault-tolerant-high-availability-reliability> [Accessed: May 2011]