FP7 Information & Communication Technologies (ICT)

CONSERN

Deliverable D2.1

System Level Energy Optimization Solutions

Document Number:	INFSO-ICT-257542/CONSERN /WP2/D2.1/31.05.2011
Contractual Date of Delivery:	31/05/2011yyyy
Authors:	Tim Lewis (Editor). See contributors table.
Workpackage:	WP2
Distribution / Type:	PU
Version:	1.0
Total Number of Pages:	61
File:	CONSERN_D2.1_v1.0.docx

Abstract: This deliverable covers energy awareness mechanisms from a system-level perspective. It explores four techniques: Petri-net models for energy simulation, system idle time estimation, non-intrusive aggregation algorithms and a real world testbed for measuring sensor battery discharge characteristics. All techniques report on the scope for energy optimisation.

Executive Summary

The Cooperative and Self-growing Energy-aware Networks (CONSERN) project [1] is an ambitious FP7 EC Specific Targeted Research Project (STREP) aiming at developing and validating a novel paradigm for dedicated, purpose-driven small scale wireless networks and systems, characterized by a service-centric evolutionary approach, introduced here as an energy-aware self-growing network.

This deliverable D2.1 is the first deliverable of work package WP2 and is provided together with D2.2 that is written in parallel. Some issues of the first and second milestones in WP2 are again outlined here with a more concrete and precise view on system level energy optimization. This deliverable reports some low energy protocol discussions and the introduction of system and terminal level energy optimization aspects already made in internal milestones within this work package.

This deliverable follows three techniques for assessing energy optimisation of systems and networks.

The first two use theoretical modelling and simulation techniques:

- Petri-net modelling used to quantify the benefits of a decentralised sensor network system.
- System Idle Time Estimation techniques used to explore a couple of different power level selection strategies.

The third technique approaches the problem from a more practical testbed direction, where a sensor Network testbed is used to dynamically collect energy consumption data from active sensor nodes. This can compare various strategies to discover the level of energy saving that they provide.

The main findings of this CONSERN work on system level energy consumption are:

- The Petri-net simulation technique allows models of centralised and decentralised sensor polling applications to be developed. For sensible values of global and local transmission power levels, there is benefit from a switch from centralised to decentralised mode at networks of size around ten to twenty nodes, sizes which are appropriate for the home networking case that has been targeted.
- The System Idle Time estimation approach looked at two strategies of power level selection, and found trade-off points based on different system parameters (such as the number of power states), and a desired system property metric based on successful event detection rates and event energy consumption percentages. Results from the instrumentation tool show that the energy gains derived from more intelligent sleep strategies has the potential to be quite significant.
- Real-life testbed implementation of the Non-Intrusive Aggregation algorithm demonstrated how the number of packet transmissions could be dropped using a range of different levels of aggregation. In turn this reduction in packet rate is interpreted in terms of energy reduction in the system, with energy reduction of between 30-50% depending on the type of aggregation.
- Measurement work on another real-life testbed implementation found that useful discharge characteristics of the sensor node battery could be obtained over time, and predictions of when batteries reach critical state could be made. Some evidence of battery recovery when devices were left idle was found. The discharge characteristics could also be measured from the perspective of packet transmission numbers, allowing for the possibility of optimising by rationing packet transmissions over a particular period.

Document Revision History

Date	Version	Author	Summary of changes
21-01-2011	0.01	Evangelos Rekkas (NKUA), Andreas Merentitis (NKUA)	Initial ToC
28-3-2011	0.02	MS, TL	Added contributions from Fraunhofer and TREL
29-3-2011	0.03	Opher Yaron	Added IBBT contribution
30-3-2011	0.04	Athanasios Makris	Added NKUA contribution
15-5-2011	0.1	Opher Yaron	IBBT contribution
16-5-2011	0.2	Tim Lewis	Integrated NKUA, IBBT contributions
23-5-2011	0.2	Mario Schuster	Contribution related to energy awareness in wireless sensor networks
25-5-2011	0.3	Mario Schuster	Further input on practical measurements of battery status of sensors as well as interface description
26-5-2011	0.5	Tim Lewis	Restructure, add editorial sections
1-6-2011	0.6	Tim Lewis	Integrate final updates from NKUA, IBBT, Fraunhofer
7-6-2011	0.7	Alexandros Kaloxylos	Additional contribution
8-6-2011	0.8	Tim Lewis	Update of objectives table, editorial changes
17-6-2011	1.0	Tim Lewis	Incorporation of review revisions

Contributors

First Name	Last Name	Company	Email
Evangelos	Rekkas	NKUA	erekkas@di.uoa.gr
Andreas	Merentitis	NKUA	amer@di.uoa.gr
Mario	Schuster	Fraunhofer	mario.schuster@fokus.fraunhofer.de
Tim	Lewis	TREL	tim.lewis@toshiba-trel.com
Opher	Yaron	IBBT	opher.yaron@intec.ugent.be
Eli	De Poorter	IBBT	Eli.DePoortereli.depoorter@intec.ugent.be
Athanasios	Makris	NKUA	tmakris@di.uoa.gr
Alexandros	Kaloxylos	NKUA	agk@di.uoa.gr

Acronyms

Acronym	Meaning
6LoWPAN	IPv6 over Low Power Wireless Personal Area Networks
ACM	Association for Computing Machinery
AM	Active Mode
AP	Access Point
API	Application Program Interfaces
ARM	Advanced RISC Machine (processor architecture)
CC	Cycle Callable
CCE	CONSERN Cognitive Engine
CMP	Chip Multi-processors
CONSERN	Cooperative and Self-growing Energy-aware Networks
CR	Cognitive Radio
DoW	Description of Work
DT	Design Time
DTIM	Delivery Traffic Identification Message
DVFS	Dynamic Voltage and Frequency Scaling
FDT	Formal Design Techniques
FIFO	First In First Out (queuing discipline)
FU	Function Units
GB1	Go-Back-1 Model
GBH	Go-Back-Half Model
IEEE	Institute of Electrical and Electronics Engineers
INDRA	Information Driven Architecture
IP(v6)	Internet Protocol (version 6)
IPFIX	IP Flow Information Export
IRMS	Electric Current (Intensity) Root Mean Square
LEACH	Low Energy Adaptive Clustering Hierarchy
LSC	Live Sequence Charts
LTL	Linear Temporal Logic
MAC	Medium Access Control
MIMO	Multiple Input Multiple Output
MINT	MIPS Emulator
MIPS	Microprocessor without Interlocked Pipeline Stages (processor architecture)
MIT	Massachusetts Institute of Technology
MMSE	Minimum mean square error
NETCONF	Network Configuration protocol
NoC	Network on Chip
PEPA	Performance Evaluation Process Algebra
PHY	Physical Layer
PIM	Processor-in-Memory
PIPE	Platform Independent Petri-net Editor
PROMELA	PROMELA Modelling Language

Acronym	Meaning
PS	Power Save (mode)
PV	Programmer's View
QoS	Quality of Service
RENEW	Reference Net Workshop
RFIC	Radio Frequency Integrated Circuit
RT	Run Time
RTL	Register Transfer Level
SEP	Success-Energy Product
SESC	SuperESCalar Simulator
SIC	Successive Interference Cancellation
SIG	Special Interest Group
S-Invariant	Place invariant
SNMP	Simple Network Management Protocol
SNR	Signal-to-Noise Ratio
SoC	System-on-chip
SPIN	Sensor Protocols for Information via Negotiation or SPIN model checker
SPN	Stochastic Petri-net
STA	Wireless Station
STEM	Sparse Topology and Energy Management
TF	Timed Functional
TIM	Traffic Identification Map
T-Invariant	Transition invariant
TLM	Transaction Level Modelling
UC	Use Case
UML	Unified Modelling Language
VRMS	Voltage Root Mean Square
Wi-Fi	Trademark of Wi-Fi Alliance (WLAN certification)
WLAN	Wireless Local Area Network
WP	Work package
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Table of Contents

List of Figures

List of Tables

1. Introduction

This deliverable, together with Deliverable 2.2 [23], are the first deliverables of Work-package 2. The scope of WP2 is discussed in more detail in D2.2, as are the main technical objectives of the work package and how they link to the WP2 deliverables.

Section 1.1 focuses on the scope of Task 2.1 as this deliverable presents the mechanisms and results that are currently achieved in this task.

To situate this deliverable's technical contributions in the wider scope of the CONSERN project, Section 1.3 looks at the links with the other work packages. The use-cases derived in deliverable D1.1 are considered in section 1.3.1 by selecting the use cases that are couple with the scope of the deliverable. Links with work packages 3 (Section 1.3.2), 4 (Section 1.3.3) and 5 (Section 1.3.4) are then covered.

1.1 Work-package and Task Scope

Figure 1-1 shows the structure of WP2. Firstly there is a split between energy aware techniques that can be used at run-time (dealt with later in Task 2.3) and those that can be applied during the network design phase. Secondly, techniques that are relevant at the device or terminal level are dealt with in Task 2.2 (and Deliverable 2.2), whilst Task 2.1, (and this Deliverable) deal with optimisations at the system level. At this stage of the CONSERN project the distinction between terminal and system is important, when we later turn to run-time energy optimization techniques, we will need to combine these approaches to build self-adaptive and self-calibration schemes.

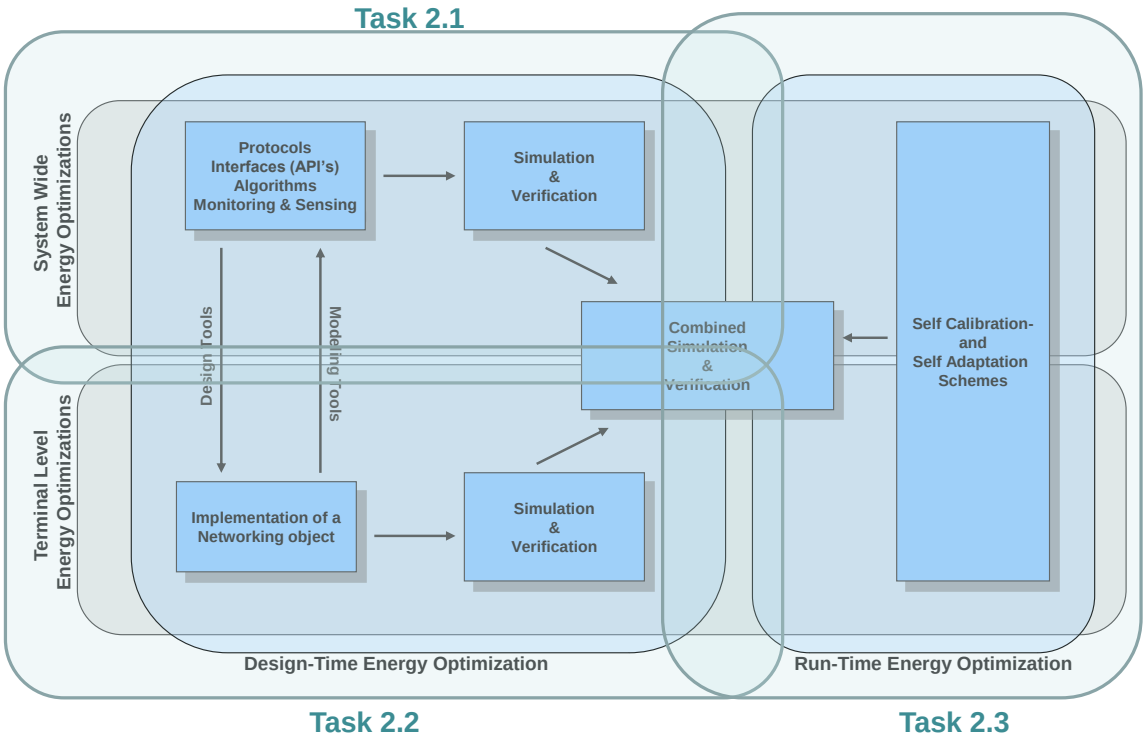

Figure 1-1: Structure of WP2.

Table 1-1 shows the technical objectives of Work-package 2, with the ones of specific relevance to Task 2.1 shown shaded. The extent to which these objectives are covered by this deliverable is outlined in the table. Note that although objective 5 has been assigned as a Task 2.2 objective, there

is some coverage of its goals in this document. Also note that objectives 6, 7 and 8 relate to the later Task 2.3 into which all this work will feed.

Table 1-1 Coverage of Work-package 2 Objectives. The shaded ones are covered by Task 2.1.

Objective	Objective Description	Coverage within D2.1
1	Evaluation and selection of appropriate low-power protocols;	802.15.4/ZigBee and IPv6/6LoWPAN are identified as appropriate low power protocols for the sensor platform in Section 3.3.
2	Modelling tools for algorithmic solutions,	Modelling tools are described in Section 4.1.
3	Design and optimisation of protocols for energy efficiency;	The Petri-net framework Section 4.1.1 allows modelling of energy consumption, as does the System Idle Estimation (Section 4.1.2) and the Non-Intrusive Aggregation technique (Section 3.2).
4	Interfaces and data structures towards measurement and control data for energy optimisation purposes;	Section 2.1 covers interfaces between sensor networks and cognitive entities. Section 5.4 shows how these can be used to collect energy consumption data.
5	Development of tools for modelling and management of platform energy consumption in a 'real-link' context,	The IDRA platform 3.2 promises management of platform energy consumption for a real system.
6	Design tools targeted towards energy efficiency;	The System Idle Estimation (Section 4.1.2) together with Binary Instrumentation (Section 4.1.2.3), Petri-net modelling are both powerful design tools to promote energy efficiency.
7	Development of methods which allow monitoring and adapting the energy consumption of wireless devices radio platforms, resulting in considerable average power savings,	Monitoring and (eventually adaptation) of consumption is performed in the Sensor Network platform (Section 4.2.1). The System Idle Estimation work allows for adapting energy consumption as do the Non-Intrusive Aggregation techniques (Section 3.2), which reports significant savings of 30%-50%.
8	Coupling of energy optimisation approaches at system and terminal level,	These two objectives will be dealt with in Task 2.3, into which the work of Tasks 2.1 and 2.2 feeds.
9	Run-time calibration and self-adaptation schemes for low energy through network awareness.	

1.2 The Rest of the Deliverable

This deliverable follows three techniques for assessing energy optimisation of systems and networks.

The first two use theoretical modelling and simulation techniques:

- Petri-net modelling used to quantify the benefits of a decentralised sensor network system,
- System Idle Time Estimation techniques used to explore a couple of different power level selection strategies.

The third technique approaches the problem from a more practical testbed direction, where a sensor network testbed is used to dynamically collect energy consumption data from active sensor nodes.

The remainder of the deliverable describes further the main technical contributions of Task 2.1. Section 2 considers the interfaces to the cognitive decision making required for energy optimisation. Section 3 describes the four mechanisms for energy reduction that are covered in this task. Section 4 looks at the tools, approaches and testbeds that are used to validate and quantify these mechanisms.

Section 5 reports on the results that we obtained from the various techniques, and Section 6 concludes with a brief summary of the findings of this part of the task.

1.3 Links with other Work Packages

This section looks at the other work package contributions (WP1, WP3, WP4 and WP5) from the perspective of work package 2, particularly considering the system aspects.

1.3.1 WP1: Use-case refinement

Table 1-2 shows the project use-cases (from Deliverable 1.1 [22]) that deal with system level energy optimization and are thus relevant to this deliverable.

Note that use-cases contributed by partners who are not directly involved with this deliverable are greyed.

Table 1-2: Use-cases relevant to Task 2.1 and this Deliverable

Use Case	Partner	Title
UC-01	OTE	Energy Optimization in a moving vehicle with capacity and coverage limits.
UC-02	NKUA	Energy Optimization in an Office environment under coverage constraints.
UC-03	NKUA	Energy Optimization for Self-Growing Office environment under coverage and capacity constraints.
UC-5	HWSE	Switch on-off of nodes for Energy Efficiency in Heterogeneous Networks
UC-6	HWSE	Cooperative DAS nodes configuration.
UC-7	HWSE	Cooperative relay for Energy Efficiency.
UC-08	Fraunhofer	Energy-aware end-to-end delay optimization.
UC-09	Fraunhofer	Purpose-driven network configuration during an emergency situation.
UC-11	IBBT	Energy optimization of co-located wireless networks in a home/office environment.
UC-13	TREL	Home Monitoring Energy Optimization.

1.3.2 WP3: Cooperation and Collaboration Mechanisms

This section summarizes work in the cooperation and collaboration mechanisms work-package (WP3), from the perspective of system-level energy concerns.

D3.1 [24] identifies a number of enabling technologies for energy efficient networking, including network coding, cooperative relay, channel selection and collaborative radio control mechanisms. The more general problems of cooperative systems are tackled using information fusion techniques. D3.2 [25] develops a number of system models, including models of cooperative network coding, and cooperative power and network control.

Cooperation and collaboration mechanisms involve systems concepts almost by definition, since they require sets of independent actors. In fact M3.2 [26] explicitly covers the interaction between work-packages WP2 and WP3, so this section will briefly summarise that report, with specific highlighting of the aspects of interest to the task T2.1.

There are two key concepts that cover the interaction between the two work packages, energy awareness and the balance between autonomic versus cooperative operation. From a systems perspective energy awareness comprises the examination of energy consumed throughout all levels of the communication stack. Due to the focus of the project, there are a large number of parameters of interest that relate to energy awareness; most of these are shared between the two work packages. Of those that are not, perhaps the ones of most interest are those relating to power trade-offs with either operator policies or QoS issues. Although WP2 does consider network/system issues (in this very deliverable), the direction of WP2 is more towards the operations inherent in a network node, whilst WP3 focuses more on the information exchange for co-operative operation.

The balance between autonomic and cooperative behaviour can, itself be approached from two different directions. Firstly, there is a trade-off between the individual objectives of a device and the objectives of the network. Secondly the optimal network configuration depends on the balance between fully cooperative methods and simpler independent (or autonomous) decisions.

1.3.3 WP4: Enablers for Self-growing Paradigms

Work package WP4 works on enablers for self-growing paradigms which are described in the deliverables D4.1: “Initial Description of Self-Growing Scenarios, Properties, Requirements and Envisaged Framework” (month M06, [27]) and D4.2: “Distributed Self-Growing Architecture and Interface Description” (month M11, [28]). The deliverable D4.1 introduces the basic concepts behind the self-growing paradigm of CONSERN, describes a methodology for that, and elaborates on the CONSERN use cases (defined in D1.1, [22]) from the self-growing perspective. D4.2 provides formal specifications for the functional entities based on the requirements and outcome of T4.1/D4.1. It also focuses on a dynamic cognitive control architecture and defines formal interfaces.

A self-growing network may have also a big impact on energy optimization issues. The related changes on specific parameters due to changes in the current network configuration based on events or incidents that require “self-growingness” should be considered at system and terminal level as well. Especially, Self-X aspects as explained in D4.1 may have a direct effect on the energy consumption of a network, a part of a network, or a single node/system or terminal due to increased communication or a general new goal of the network that must be achieved. In other words, the idea of a purpose driven self-growing network as introduced in CONSERN has an impact to the energy optimization of the involved networking elements. This means that a system or terminal which is capable to have “self-growingness” should reserve spare energy or failback energy strategies to handle the communication overhead of unforeseen events or Self-X reconfigurations. On the other hand, the self-growing mechanisms should also consider the remaining energy of nodes and their priorities of processing certain tasks to decide whether a node can take over additional roles or tasks due to Self-X requirements.

Furthermore, the WP4 deliverable D4.2 has an impact on data structures and interfaces used between the cognitive self-growing entities and the physical devices (e.g. actuators, sensors, terminals). It introduces the so called CONSERN Cognitive Engine(s) (CCE) and Functional Unit(s) (FU) modules. Especially, the FUs are responsible for directly controlling or sensing/scanning the environment and thus sensors/actuators, terminals, and other system components must be represented by one or more FUs to enable their use within the cognitive self-growing architecture. The communication between CCEs and FUs is distributed and CORBA is preferred here as middleware. Therefore, interfaces are defined using CORBA-IDL and the data structures and interfaces defined in WP2 should consider that to let energy awareness functions a part of the overall self-growing approach.

1.3.4 WP5: Validation and Proof-of-Concept

System level energy optimization solutions designed in the scope of WP2 will be demonstrated in a series of scenarios in WP5 as presented in an internal project milestone. Table 1-3 below presents the technical contributions from WP2 and the demonstrators in which they will be implemented in WP5.

Table 1-3: WP2 requirements and mechanisms and their relation with the test bed demonstration

Partner	Mechanisms test bed application
Fraunhofer	Mechanisms for protocol and system level energy awareness and optimization. Terminal energy aspects are not directly addressed here, but the case in which a terminal device directly connects to the WSN (e.g. through one of the Wi-Fi routers) will be demonstrated.
NKUA	Mechanisms for system level energy optimization. Detailed implementation steps on how to apply these mechanisms onto their test bed have already been defined in M5.1 [29].
Imec	Development of tools for modelling and management of platform energy consumption in a “real-link” context, Design tools targeted towards energy efficiency; Development of methods which allow monitoring and adapting the energy consumption of wireless devices radio platforms. These mechanisms will also be presented in demonstrations of WP5.
Imec/IBBT	Run-time learning schemes for low energy through network awareness in WSNs. In this case the objects will not rely on a Design Time (DT) optimisation but will gradually learn and update their optimal operating points utilizing information gathered by sensing the network. There is the intention to integrate these schemes to the IBBT test bed.

2. Interfaces between Sensor Networks and Cognitive Entities

As described in the CONSERN DoW [1], WP2 also focuses on energy efficiency and optimisation at networking or system level, i.e. at a higher level of abstraction. This includes analysis of protocols and networking technologies and their energy-aware capabilities, including options for power-aware protocol optimization, PHY/MAC characteristics, network and routing characteristics, as well as higher layer and other characteristics of the protocols. Embedded systems that suffer from resource constraints are considered here in terms of system level techniques for energy optimization. Especially, task T2.1 investigates interfaces and data structures to provide access to measurement and control data, and to communicate this data via suitable protocols. This considers data related to sensors and actors instantiated by nodes residing within sensor networks as well as those residing in the access network domain and will comprise e.g. data obtained from RF sensors, environmental sensors and data traffic measurement probes associated with protocol layers.

2.1 Capabilities in Sensor Networking Systems

Sensor networking devices such as wireless sensors/actuators or other types of embedded systems often have certain capabilities, e.g. limited computational resources, they are battery powered, or limited memory. They can be used to directly measure environmental conditions that might be useful for the decision making processes of CONSERN (e.g. a wireless motion detector can be used to detect people moving in an area covered by a CONSERN infrastructure) or they can indirectly measure conditions of other CONSERN nodes (e.g. power plugs or similar sensors that can measure the energy consumption of a node).

To embed such devices in an overall CONSERN networking infrastructure capable for energy awareness and self-growing, those devices require additional functions to fulfil the related requirements or must reserve additional spare memory, processing power, or energy to be able to operate temporarily in a different role (e.g. activating a burst data transmission mode in an emergency situation). Furthermore, it would be a benefit if these devices can also support at least basic Self-X capabilities in terms of a possible recalibration of some parameters (e.g. setting RF parameters dynamically via AT commands or similar methods) or uploading code pieces or rule sets to be executed during runtime to the nodes. If this is not possible directly on the embedded system, proxy nodes with more computing power (e.g. collection nodes, cluster heads, base stations, or PC like systems) may take over this role.

2.2 General Considerations

For the cognitive elements and decision making processes (WP3, WP4), systems and terminals can provide basic attributes for energy awareness or optimization. These attributes should fit into the information models that are foreseen as base for decision making (in a best case in a generic and technology independent manner). At the moment, the structure of the related data elements is still under discussion. The idea is to map the relevant parameters for energy awareness from D1.1 to concrete data structures or elements in a message send by a certain protocol and also to cross-check with the outcome of WP4 (especially D4.2). For instance, in the sensor networking domain a binary format is preferred to reduce the transmission overhead. In the IP domain or at higher level systems, state of the art representations like XML, RDF, or OWL are also possible. In any case, a mapping between pure binary representations and high level representations is required (e.g. through gateways or proxy nodes).

Energy efficiency and optimisation at networking or protocol level can be achieved by using energy capabilities of different protocols or introduce new methods (application level protocols and functions, additional messages) to collect information about the energy capabilities and energy state of the involved nodes and to adaptively change the energy behaviour. For this purpose, a higher

level abstraction for the distributed energy information must be developed to provide decision making functions or components with a unique interface that can be the base for their rules or policies. This concept is shown in the following figure:

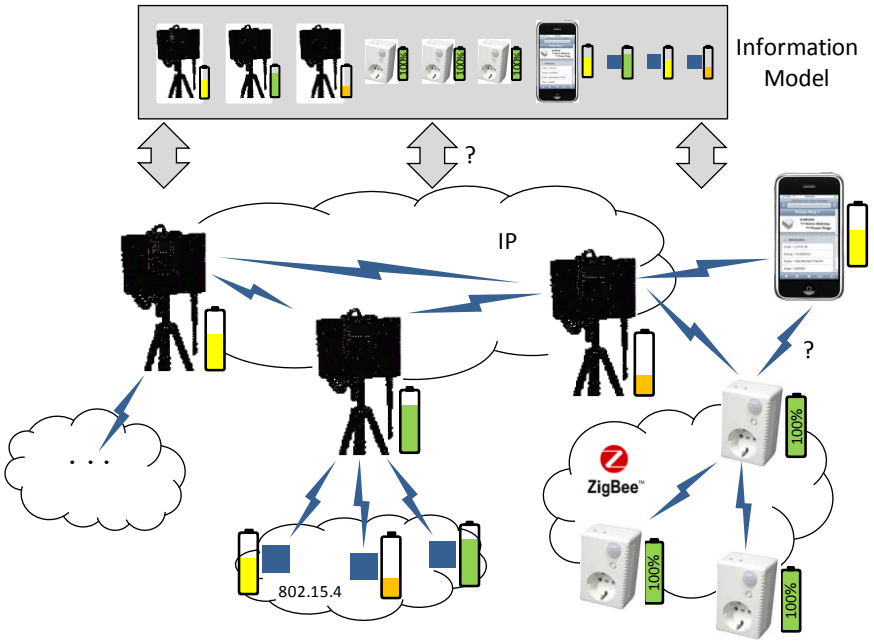

Figure 2-1: Collection of energy capabilities in a distributed environment.

It shows the many different energy capabilities of the various devices (WSNs, mobile terminals, routing equipment). The battery symbol is just a placeholder for the whole set of possible parameters for energy capabilities such as type of power (mains power, battery), energy options in the used protocols, energy functions in the device (sleep, stand-by etc.), current consumption, resource constraints, estimated battery life time and others. This kind of information must be distributed in the network and over network boundaries to get some abstract view of all network capabilities of the networking environment (that is used to collectively achieve a situation driven goal). The energy capabilities should be mapped here to a unique and abstract "information model" based on legacy protocols or concepts. Technologies like Web Services, SNMP, CORBA etc. are under discussion at the moment for achieving this goal (see testbed and interface discussions in [29]).

2.3 Data Example for the ZigBee Power Plug

One example for data elements could be the parameters of the ZigBee power plug that can be used to indirectly measure the power consumption of other nodes. These devices have the following parameters:

- VRMS in Volt, e.g. 220V,
- Load in Watt, e.g. 40W,
- Work in kWh (e.g. 4kWh if the connected 40W device runs for 100h),
- Frequency in Hertz, e.g. 50Hz,
- Status, e.g. "on" or "off".

These values are predefined by the vendor and have a related message format (proprietary string based values sent over ZigBee networking layer). Other devices such as battery powered sensor nodes (like the ZigBee multi sensor) have different parameters with more or less the same meaning. At the lower protocol level, data or message format is often given by the used technology or the

vendor. These formats must be mapped to the information model foreseen by the decision elements.

Another aspect, especially in the sensor networking domain is the integration of WSNs into existing IP environments or the direct use of IP(v6) in the WSN domain (e.g. by technologies like 6lowPAN). For instance, this integration is done through routers that have WSN base stations and Wi-Fi interfaces equipped and can forward the WSN communication (non-IP) to the IP domain. Here, some mediator protocols or technologies could be used for data representation and message forwarding based on well-known or always accepted standards. Examples for those technologies are NETCONF, IPFIX, or SNMP. The benefit is that existing devices like routers, switches, or other networking equipment often support these standards and the decision making elements build up on the top of it could have access to both worlds, the classical networking environments and the WSN domain. The direct use of these protocols on the WSN nodes often is impossible due to the limited resources of such systems and developing appropriate solutions is not one of the main goals of CONSERN. However, in the future solutions like IPFIX over 6lowPAN in 802.15.4 based sensor networks (see [46]) could be considered.

In the testbed descriptions, it is proposed (see testbed and interface discussions in [29]) o use Web Services or SNMP to provide access to sensor node information, energy capabilities, and sensor data or to actively control the sensor or actuator node (see [29]). The SNMP part is still under development, but the Web Service part is already developed and provides a state of the art high level approach that is programming language and runtime independent (the Web Service is a .NET solution, but thanks to the use of standards like WSDL and SOAP it should be accessible from other runtime environments or programming languages as well).

2.4 Self-Growing Interface Descriptions

WP4 has introduced some aspects in D4.2 that must be considered for the definition of data structures and interfaces in WP2. The CONSERN Cognitive Engine(s) (CCE) and Functional Unit(s) (FU) modules have been defined in D4.2 and at least the FU modules can be responsible for providing energy awareness functions related to sensor networks or other types of networks and enabling cognitive functions based on them. Figure 2-2 is taken from D4.2 and depicts the cooperation between the different cognitive elements including the different types of interfaces. Especially, the gateway module is a preferred candidate for providing access to sensor networks or resource constraint devices that are not able to have own cognitive functions. In this case, the gateway performs the basic communication to the other network parts and the FU that resides on the gateway can be seen as one or more proxy functions for the real sensor nodes or embedded systems behind the gateway. The concrete interface and data structure format here is still under discussion. However, WP4 has proposed CORBA and IDL as middleware and description format. That should be taken into account.

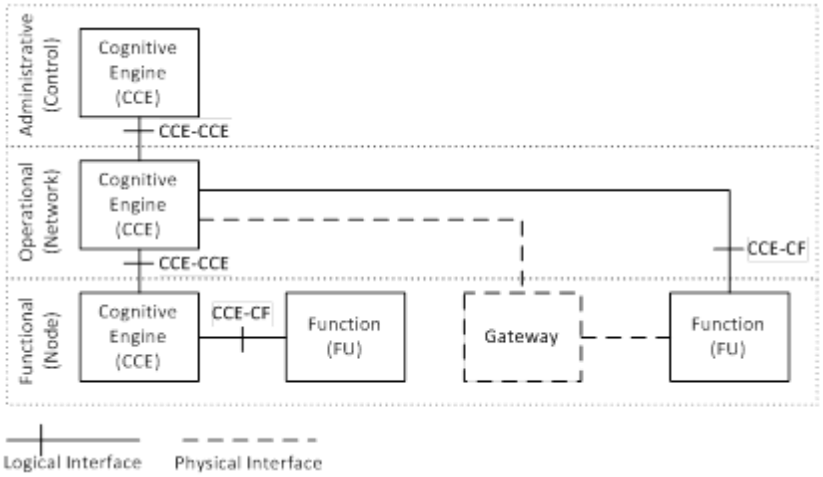

Figure 2-2: Layered logical self-growing architecture.

3. Mechanisms for System Energy Optimisation

This section looks at the various mechanisms proposed for optimisation that are explored by the three techniques. Section 3.1 looks at the use of decentralisation as an energy optimisation process, Section 3.2 considers non-intrusive packet aggregation, and Section 3.3 considers the use of existing low-power and resource optimized protocols.

3.1 Sensor Network Decentralisation Application

The chief use-case driving this part of the work is UC-13, "Home Monitoring Energy Optimization". We will develop the important aspects of the use-case here, for its full description see D1.1 Section 4.1.13. Here we describe the system model with the different centralised and decentralised modes.

A number of sensor/monitoring devices are placed in a home environment. Each sensor has available at least one measurement device that collects values of temperature, humidity or light levels. These values are reported back to a central monitoring gateway. It is assumed that the gateway device is mains powered, so in comparison to the battery powered sensor devices is unconstrained in its use of power.

In addition to measurement and transmission functionality, each sensor can also act as an aggregator, combining samples from a number of other sensors and forwarding the resulting sample onto the gateway, or another aggregator. The specific function that the aggregator performs will depend on the purpose of the network and the type of data dealt with. Examples of aggregation functionality are:

1. Perform direct averaging over samples received. This could be over time (a fixed size window) or over location (computing the mean temperature over a small set of neighbouring devices),
2. Perform variance computation – here the interest is in how much the data is moving either over time, or between neighbouring locations,
3. Perform merging of samples. This reduces load on the network by combining a number of similar valued measurements into one report. A large area where the temperature is uniform need only produce one combined report.

Irrespective of the exact functionality of the aggregation mechanism, we can abstract these from a power consumption perspective into the sequence of three operations:

1. A number of receptions of sample packets,
2. An aggregation computation,
3. A single packet transmission.

In line with the self-growing theme of CONSERN, we study how the performance of this application scales as the number of sensor nodes grows. Initially we assume that the system operates in centralised mode, and consists of a gateway plus a small number of sensor devices that report directly to the gateway. As more sensors are added to the network, some move into aggregator mode and allow other sensors to report their measurements to them directly. In full decentralised mode very few aggregators report to the gateway, most are aggregating samples from other aggregators. A report to the central gateway is referred to as a global transmission, one to a nearby aggregator is termed a local transmission. Figure 3-1 shows a network in centralised mode [left] and a larger network in decentralised mode [right]. The Sensor device contains processes that obtain sensor samples and aggregate and transmit aggregated values.

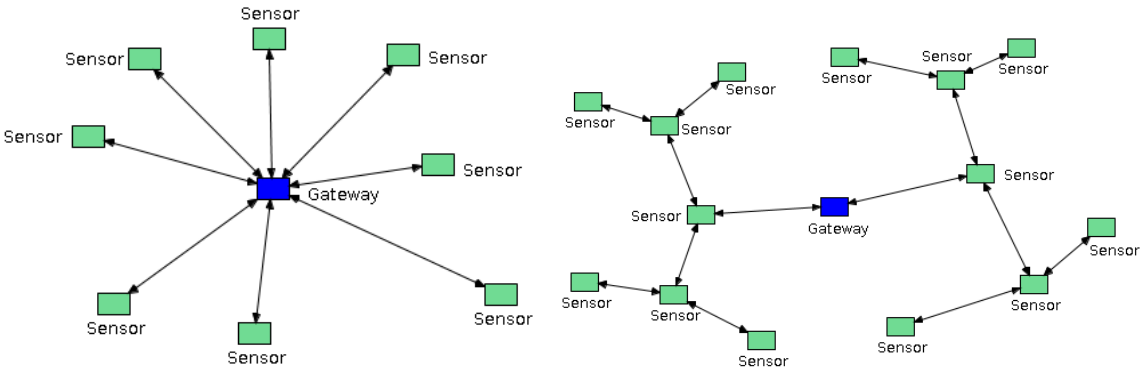

Figure 3-1: Comparison of centralised [left] and decentralised [right] modes.

The most interesting behaviour will occur at the boundary between centralised and decentralised systems. In terms of power consumption the following assumptions can be made:

1. Devices reporting locally will use lower power transmission modes to ones send to the central gateway. This is due purely to the transmission range, we assume that different power levels can be used in either case.
2. At the central gateway, computation and packet sending/reception are free in terms of power.

Thus the use of the decentralised mode of this system can act as a power saving mechanism, under certain conditions that depend on the size of the network and how densely the sensors are distributed. These conditions will be explored using the simulations explained in the next section.

3.1.1 Simulation of Sensor Network

In addition to our static power analysis we will also need to perform some network-wide power consumption estimates that may require a number of device level models to be combined.

Figure 3-2 shows the basic sensor model that is the same as the one that is analysed in D2.2 [23], Section 4.1. This device collects and transmits sensor data to its parent aggregator.

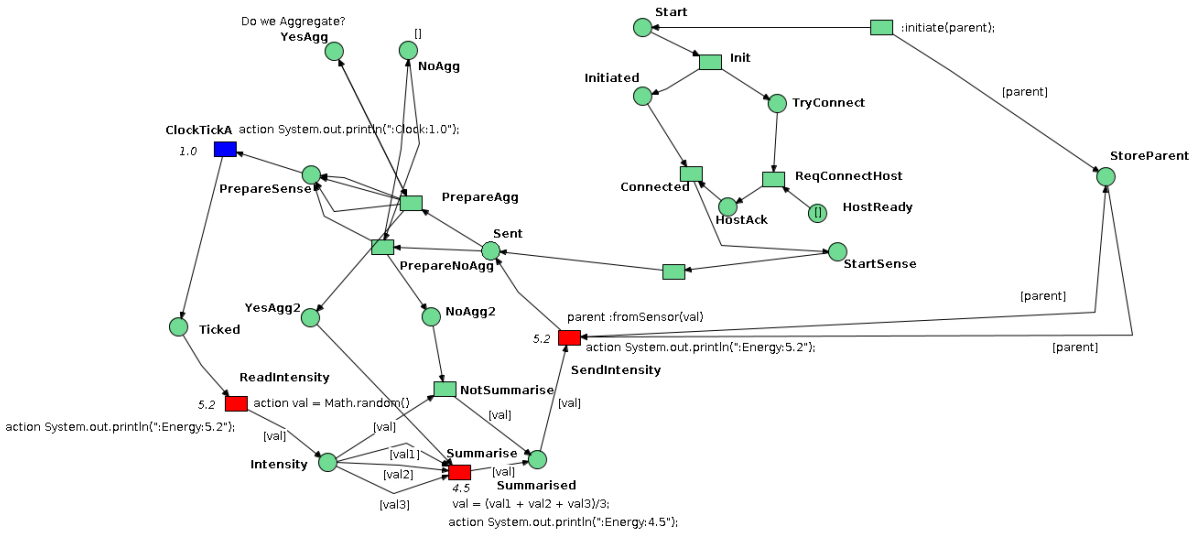

Figure 3-2: Sensor Petri-net model.

Figure 3-3 shows the aggregator functionality. A full description of the Petri-net notation and formalism is given in D2.2 Section 2.1, so is not duplicated here. This functionality can reside in a sensor or in a gateway node, we assume that every sensor has the (optional) ability to perform aggregation of data from other devices. The aggregator receives data when either the **SensorRecv** transition fires (data from its local sensor process) or **AggRecv** fires (data from another aggregator). Each aggregator process keeps track of its parent aggregator and forwards averaged data to it. These devices can then model a centralised system whereby all aggregators forward data to the same gateway aggregator, or a decentralised system whereby aggregators form a hierarchy forwarding data from the leaves back to the gateway aggregator at the root.

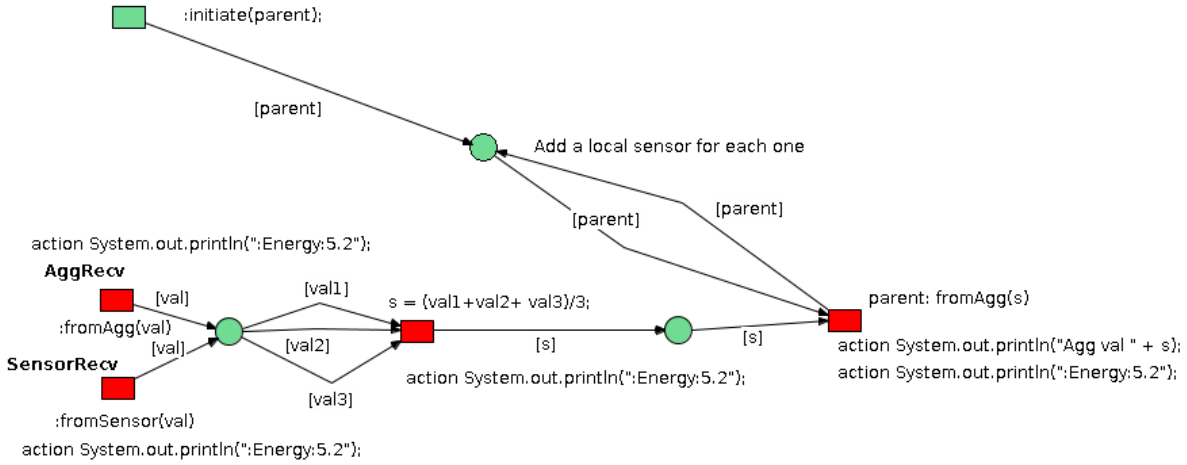

Figure 3-3: Petri-net model of Aggregator node.

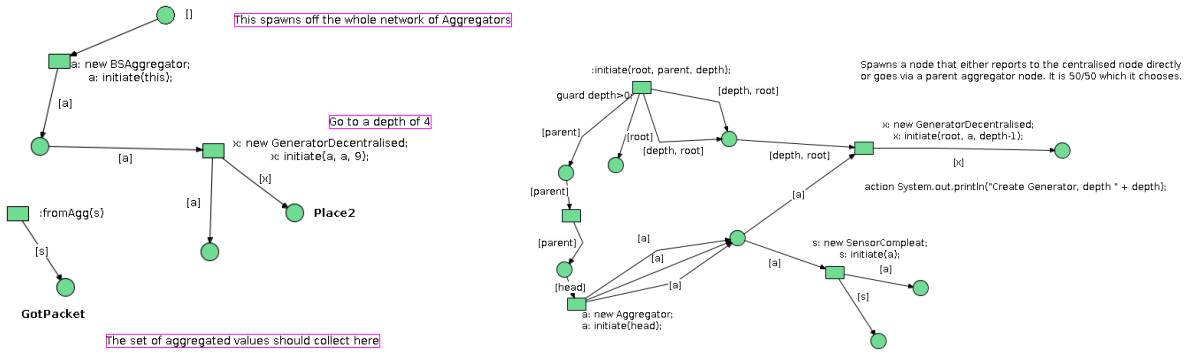

Figure 3-4: Simulation harness. The Network process [left] spawns off the set of Generators [right] that builds the system for simulation.

Figure 3-4 shows the simulation harness, consisting of Network and Generator processes. This harness is an artefact of the simulation process, and it does not relate to any physical devices in the network. It exists solely within the simulation framework to enable a number of sensor devices to be spawned and the aggregation nodes to be linked together in a meaningful way. The simulation can either work in centralised or decentralised mode, the specific models shown above generate a decentralised system, but the centralised generation works in a similar manner.

3.2 Information Driven Architecture (IDRA)

3.2.1 IDRA Description

The Information Driven Architecture (IDRA) is an adaptive architecture that supports selection per packet of which protocol to use between multiple protocols that are implemented in a network node. It is specifically designed to cope with the resource constraints and networking challenges of low-cost, low-power embedded devices that operate in heterogeneous wireless environments. In IDRA, the role of a network protocol is reduced to its main task – exchanging and processing information with other nodes. Overhead operations, such as packet creation and buffer provisioning are delegated to the IDRA system. This way, network protocol implementation is simpler and requires less memory.

When an application sends information to a remote node, for example a temperature sensor, it sends a ROOM_TEMPERATURE value to a central monitoring node, the relevant network protocol does not create a new packet. Instead, it relies on the system to send and receive information. The protocol hands the information parameter over to the system, together with the required destination. The system transparently creates a new packet and encapsulates the parameter into this packet. When the packet arrives at its final destination, the system extracts the information parameter from the packet and distributes it to the interested protocols and applications.

Packet headers are also handled by IDRA. IDRA provides a 'Packet Facade' interface that enables the network protocol to set and retrieve packet attributes, such as 'source', 'destination' and 'time-to-live'. The protocol does not require any knowledge about the actual packet structure. Instead, the packet facade is responsible for the storage and retrieval of packet attributes. Packet attributes can be interpreted by any network protocol, not only the protocol that added the attribute. To correctly store and retrieve packet attributes, the packet facade needs to know how each packet is constructed. This information is stored in packet part descriptors, which describe how and where packet attributes are stored in a header (e.g. offset, byte-ordering, number of allocated bits, etc.). Examples of packet part descriptors are an IEEE 802.15.4 header, an IEEE 802.11 Wi-Fi header or an IP header. Figure 3-5 illustrates the concept of the Packet Façade.

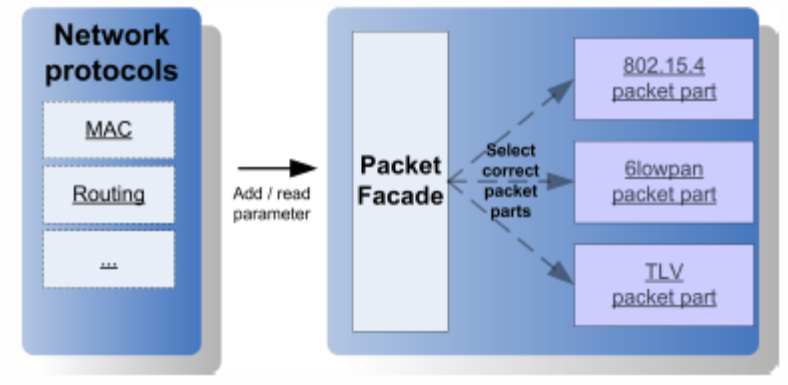

Figure 3-5: The IDRA Packet Façade.

Packet types are created by combining multiple packet part descriptors. A packet type is defined as a unique sequence of one or more well-defined packet part descriptors. For example, an IEEE 802.15.4 packet part descriptor can be combined with a 6lowpan packet part descriptor to create IPv6 compatible packets. To create new packet types, developers can combine existing packet part descriptors, or develop new propriety packet part descriptors. Alternatively, a network designer can design a single highly optimized packet part descriptor that efficiently compresses all packet attributes that are used in his specific network scenario. Finally, it is important to note that network protocols are not limited to the use of (standardized) packet attributes. Packet attributes that are

not recognized by any of the packet part descriptors are stored sequentially in the payload using a type-length-value (TLV) representation.

Finally, IDRA system is also responsible for storing created and incoming packets. Traditional, layered network protocols typically use a `store-and-process' approach, wherein each network layer stores its own packets, or a queue of pointers into the shared memory of packet buffers (Figure 3-6a). Each protocol requires a large enough internal queue to ensure that all received packets can be stored. Thus, the total amount of buffer memory increases linearly with the number of protocol layers. In IDRA, arriving packets are stored once in a system-wide shared queue, and remain there until processing is finished (Figure 3-6b).

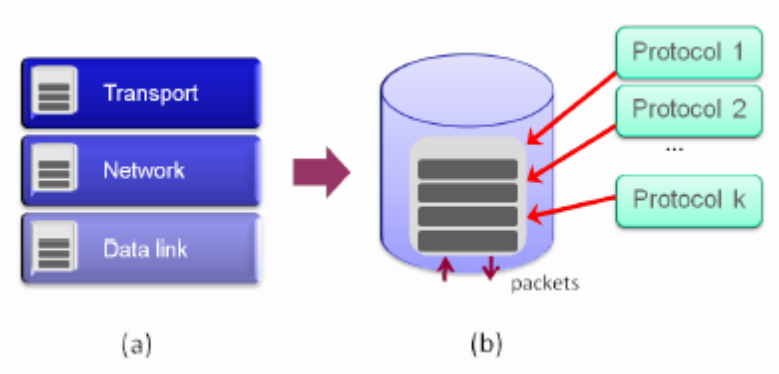

Figure 3-6: Packet Queues in a traditional layered architecture (a); and in IDRA (b).

The use of a shared, system-managed queue has several advantages: (i) protocol implementations are simpler and smaller since they do not have to allocate queue memory; (ii) packets do not need to be copied between protocols, resulting in less processing overhead; (iii) since the queue occupation from all protocols is averaged, less total queue memory is required; and (iv) monitoring and managing the total number of packets in the system is simpler.

3.2.2 IDRA Implementation

IDRA is implemented using the TinyOS operating system [30]. Run-time addition of protocols is currently not supported, since TinyOS does not support dynamic code updates. A system diagram of IDRA implementation is shown in Figure 3-7.

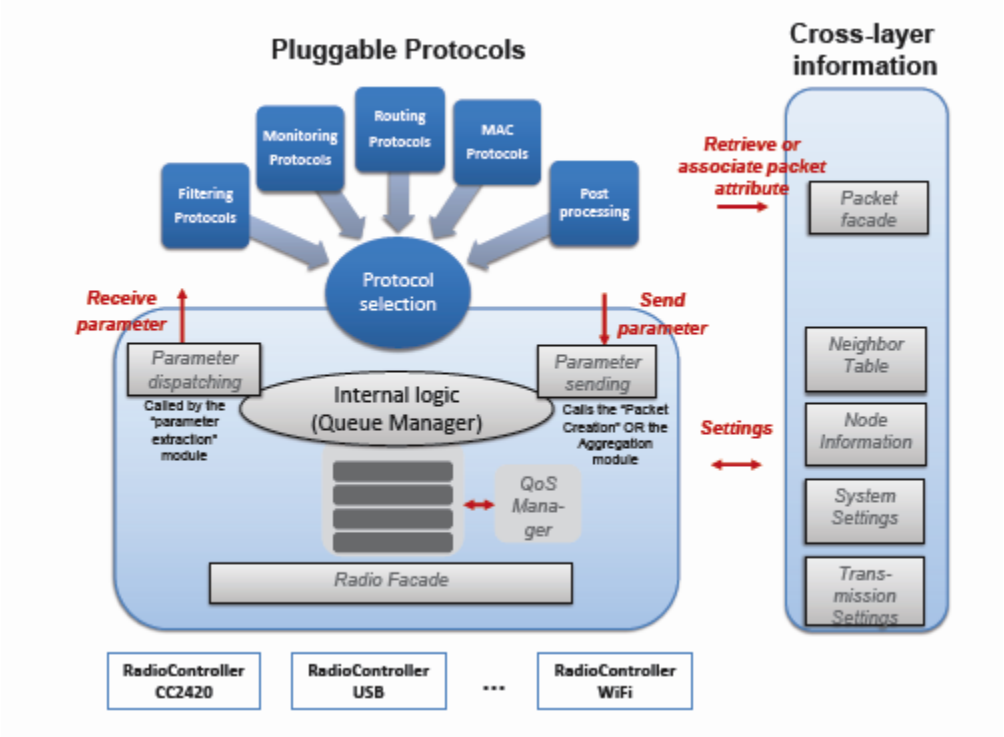

Figure 3-7: IDRA Implementation.

IDRA implementation has the following internal components.

Parameter sending / dispatching: through this component, protocols and applications can distribute information parameters to other nodes, and receive information parameters from other nodes. When information parameters do not need to be sent immediately, they are temporarily stored in an internal parameter queue so that they can be added to forwarded packets.

Internal logic: this component manages the packet queue. For each packet, status information is stored that indicates which protocols already processed the packet, whether or not the packet is ready for sending, etc.

QoS manager: at any moment, the QoS manager can drop a packet from the queue, change the priority of a packet, or indicate to the system which packet should be processed or transmitted next.

Protocol Selection: this component is responsible for selecting which network protocols should process each packet. Protocols add filters to this component to indicate for which packet types the protocol is optimized.

Packet Facade: the packet facade is used by network protocols to interact with packets and by the 'parameter sending' component to create new packets (whenever the acceptable delay of a stored information parameter has been exceeded).

Neighbor table: network protocols can use this table to store information about neighboring devices, such as link quality, battery status, etc.

Node information: this table is used as a general purpose 'whiteboard' where network protocols can store and retrieve status information.

System Settings: through this component, network protocols can read system information, such as the total number of transmitted or received packets, or update system settings, such as the node ID.

Transmission Settings: through this component, MAC protocols manage the sending of packets. It has provisions for (i) requesting how many packets from the shared queue are ready for sending, (ii) ordering the system to send a specific packet, and (iii) changing the radio settings.

Packets are processed by IDRA in four steps, as follows.

1. Storing incoming packets,
2. Selecting the next packet for processing,
3. Processing the selected packet,
4. Sending the packet,

The following paragraphs describe these four steps in detail.

3.2.2.1 Storing Incoming Packets

To limit the number of copy operations, the radio controllers can store incoming packets directly into the IDRA queue. To this end, the radio controller asks the radio facade at which queue entry the next incoming packet should be stored. When a packet is copied to the queue, the radio facade is notified and provides the radio controller with a new queue entry to store the next incoming packet. After updating the packet metadata of the newly stored packet (RSSI, LQI, packet status), the radio facade indicates to the queue manager that the packet is 'ready for processing'.

3.2.2.2 Selecting a Packet for Processing

This step starts when a new packet is stored in the shared queue. This occurs when either (i) the queue manager (Internal Logic) is notified by the radio facade that a new packet has arrived, or (ii) when Parameter Sending creates a new packet to encapsulate an information parameter. If multiple packets are available, the queue manager is responsible for selecting which packet should be processed first. Unless otherwise notified by the QoS manager, the packet with highest priority is selected first for processing. If multiple packets with the same priority are present, precedence is given to the packet that arrived first.

3.2.2.3 Processing Packets

Next, Protocol selection selects which protocols should process the packet. A single packet may need to be processed by multiple protocols. IDRA protocols can implement complex network algorithms (such as routing protocols), as well as simple modules that require no communication with other devices (such as duplicate detection). IDRA does not differentiate between protocols, except for the order at which they are executed. Complex protocols as well as simple network functions register to Protocol selection. The order at which protocols are executed is illustrated in Figure 3-8.

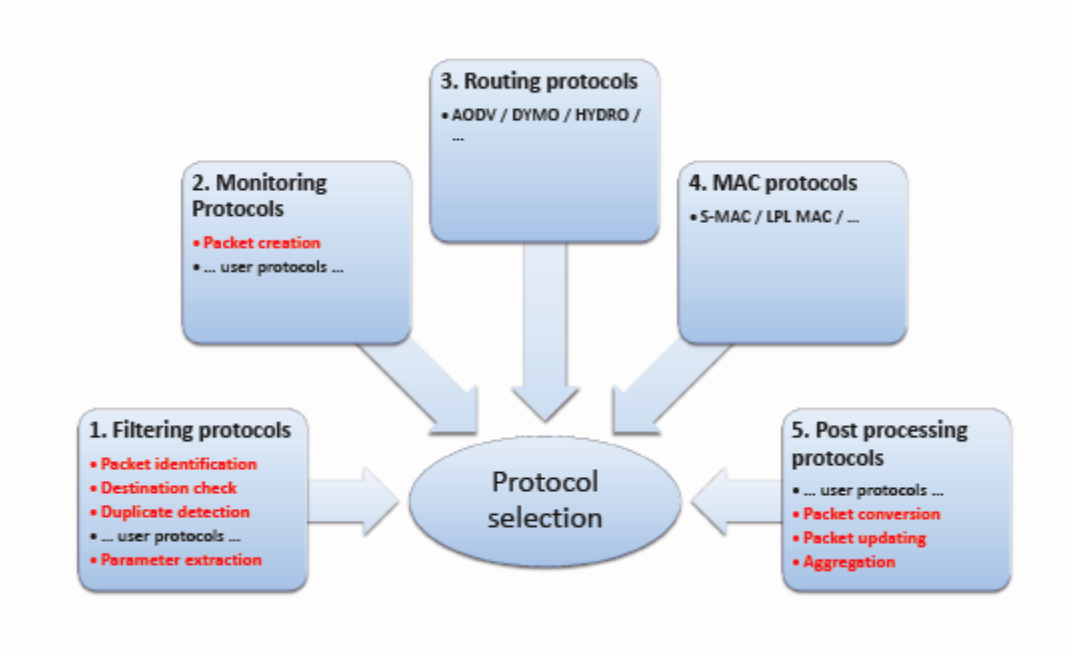

Figure 3-8: Protocol Execution Order.

1. First, filtering protocols are executed to filter away unwanted incoming packets. All filtering protocols are executed sequentially. The following filtering protocols are available.
    - **Packet identification** is responsible for identifying the packet type of incoming packets (see Section 4.5.1 of Chapter 4). Once the packet type is identified, the packet facade can be used to interact with the packet. If a packet is not recognized, it is dropped.
    - **Destination check** drops all packets that are destined to other nodes.
    - **Duplicate detection** drops duplicate packets.
    - Other user-created filtering protocols can be added. For example, **Link quality** protocols can be implemented to only accept packets from nodes that have a good link quality, or **Topology control** protocols that restrict from which neighbor nodes packets can be received.
    - Finally, **Parameter extraction** is executed. If the packet reached its final destination, the encapsulated information parameters are extracted and distributed through **Parameter dispatching**. Afterwards, the packet is dropped unless it is destined to the BROADCAST or the NEIGHBOR addresses.
2. Next, all relevant monitoring protocols are executed. Monitoring protocols typically gather network information (for example, to fill in the neighbor table). All monitoring protocols are executed sequentially. When a new packet is created by **Parameter sending**, execution starts from **Packet creation**. Thus, newly created packets will not be dropped by any filtering protocols, but will be inspected by all monitoring protocols.
3. A routing protocol is used to determine the next hop address of the packet. The packet facade is used to update the next hop address or to add a path label to the packet. To set-up a network path, routing protocols typically send out 'route request' information parameters through the **Parameter exchange** component. If multiple routing protocols are available, only one will be selected.

4. Similarly, only one MAC protocol is executed. The MAC protocol can update packet attributes, but cannot instruct IDRA to send the packet. Sending will be possible afterwards through **Transmission Settings**, once the packet has completed processing.
5. Finally, the post processing protocols prepare the packet for sending.
    - **Packet conversion** can convert the packet to a different packet type.
    - **Packet updating** updates the 'sender' packet attribute.
    - **Aggregation** checks if information parameters are available in **Parameter sending**. If possible, such parameters are aggregated to the packet.

When a network protocol finishes processing a packet, it provides a return value:

1. SUCCESS: The network protocol finished successfully; the next protocol can be executed.
2. FAIL: The network protocol cannot process the packet; the packet should be dropped from the shared queue.
3. BUSY: The network protocol is not ready yet to process the packet; the protocol will be called again at a later time.

The status of the packet is updated and the queue manager chooses which packet to process next. When **Protocol selection** indicates that no more protocols remain to process the packet, the status of this packet is updated to 'ready for sending'. If multiple packets are available for sending, the QoS manager can indicate which packet should be transmitted first.

Applications also register to **Protocol selection**. However, since applications do not process any packets, they are not part of the packet processing flow. It is possible for a protocol to register itself multiple times. For example, a MAC protocol can register itself to be executed both as a filtering protocol (to drop unwanted incoming packets) and as a monitoring protocol (to send packet acknowledges). Finally, by reconfiguring **Protocol selection**, users can change the sequence of protocol execution.

3.2.2.4 Sending Packets

The 'ready for sending' packet is stored until the MAC protocol uses the 'Transmission Settings' to indicate when and how (e.g. transmission power, radio channel) the packet should be transmitted. The 'Radio Facade' selects which radio interfaces should be used to transmit the packet and forwards the packet to the selected radio controllers.

3.2.3 Non-intrusive Aggregation

Non-Intrusive Aggregation is a mechanism to consolidate multiple packets into one, which is independent of the underlying network protocols, independent of the information sources, and is easy-to-use. It facilitates power saving by reducing the number of packet transmissions. A fundamental mechanism for power saving in wireless sensor networks is to use sleep schemes which alternately turn on and of the radio interfaces of the nodes. The idea behind this mechanism is that the radio is by far the biggest energy consumer in a wireless sensor node, as can be seen for example in Figure 3-9 , which illustrates the typical power consumption of the TmoteSky wireless sensor node [47]. However, to guarantee that the radio can be switched of regularly, the number of packet transmissions must be low. To this end, data-aggregation protocols have been proposed which combine information from multiple sources in a single packet at intermediate nodes. Since fewer packets need to be sent, the radio can be switched more often to the power-saving sleep mode. Moreover, since the number of packets is reduced, the total amount of interference and network contention is decreased.

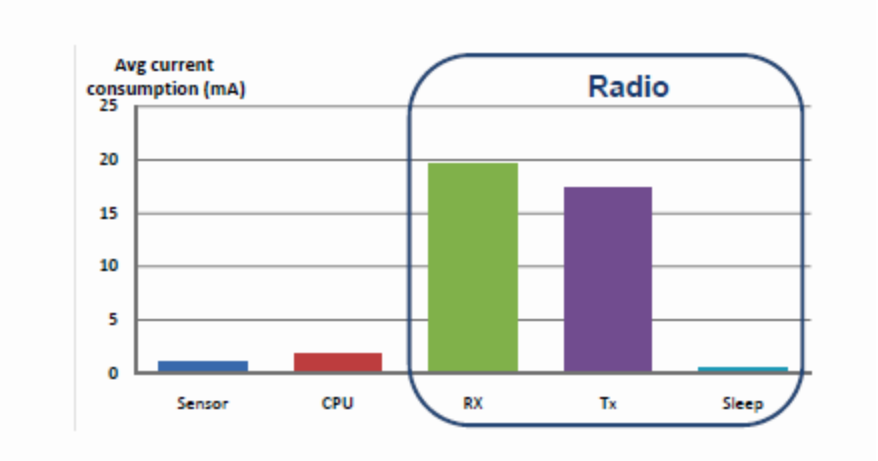

Figure 3-9: Power consumption distribution in TmoteSky.

Non-Intrusive Aggregation is implemented within the IDRA framework. The algorithm works as follows.

- For each incoming packet, the architecture will first extract all parameters which are destined for this intermediate node. The extracted parameters are distributed through the information management system to all registered protocols and applications.
- If the packet contains no more parameters in the payload, it is dropped by the system. Otherwise, the routing protocol processes the packet and sets the next hop address of the packet.
- The system checks if the destination of any of the parameters from the waiting space corresponds to the destination address or the next hop address of the packet. All matching control parameters are added sequentially to the payload of the relayed packet, starting with the parameter with the nearest deadline. This process continues until the maximum packet size is reached.

A similar algorithm is executed when the maximum delay of any of the parameters from the waiting space is reached. First, a new packet is created which encapsulates the corresponding parameter. This packet is then routed to the correct next hop address. This way, the system ensures that time-sensitive protocol parameters are delivered timely.

The packet facade of IDRA is used by the information management system to create a new packet when a parameter deadline has expired, and to retrieve information such as `source', `destination', `QoS ID' or `time-to-live' from passing packets. The packet facade can interpret any packet type, thus it ensures that the aggregation mechanism can request the next hop and destination address of any packet that passes through the system. As a result, our implementation does not require any specific knowledge about the format of relayed packets. This way, it is not only protocol-independent, but also packet-independent.

3.3 Low Power and Resource Optimized Protocols

Some existing protocols are optimized for limited and resource constrained devices as well as for saving energy by using special hardware modes (e.g. stand by, idle modes, switching system (RF) parts of etc.) or using smart protocol strategies to reduce the traffic, the send intervals, or the signal strengths of the nodes in a wireless sensor networks. Dependent on the sensor networking technologies available in the testbeds of the partners, we just introduce the 802.15.4/ZigBee family of devices and give a short introduction of IP(v6)/6LoWPAN as candidates for interconnection of WSNs (IP(v6)) or direct use in sensor networks (6LoWPAN).

802.15.4 and ZigBee

The standard IEEE 802.15.4 [41] specifies physical (PHY) and media access control (MAC) layers for so called low-rate wireless personal area networks (LR-WPANs). Other higher layer protocols like ZigBee [42], WirelessHART [45], or 6LoWPAN [43] can use 802.15.4 for the underlying PHY/MAC layers. Due to the general design approach of the standard for low power short range communication, lower energy consumption is more or less a build-in feature (e.g. compared with WiFi or Bluetooth). However, other technologies in the 2.4GHz (again WiFi or Bluetooth) or the type of data that is transmitted can increase the energy consumption due to a lot of retransmissions or bandwidth increase. WP2 considers the different options (e.g. sleep modes, duty cycles, change of transmission power, optimized topologies and channel selection). One big question here, is whether the related 802.15.4 boards supports on demand runtime configuration of these parameters. For instance, a typical module such as the XBee (PRO/ZNET) modules from DIGI allows the setting of some parameters through AT commands. WP2 also considers protocol techniques or message types that can be used to manage these parameters (e.g. by passing or tunnelling local or remote AT commands to the modules).

IP(v6)/6LoWPAN

One of the major challenges is the change from IPv4 to IPv6. The self-growing and energy aware approaches of CONSERN should be adaptable to an overall IPv6 based infrastructure and this should also include wireless sensor networks (WSN) or other classes of WPANs. Here, a direct integration of IPv6 on low power modules or embedded hardware is often not possible or requires too much computing resources or energy. Alternative approaches like 6LoWPAN [43] use some techniques like header compression and address mapping to reduce the overhead for sending IP packets (e.g. over 802.15.4). The benefit is that IP based routing protocols can be used (e.g. ROLL, LOAD, HiLow) and all nodes in the WSN are addressable through IPv6 addresses.

4. Tools for System Energy Optimization

This section considers the two main CONSERN approaches for handling system-level energy optimisation, firstly modelling and simulations frameworks (Section 4.1), and secondly approaches that use real testbeds and experimental platforms (Section 4.2).

4.1 Modelling and Simulation Frameworks

This section provides a description of the modelling and simulation tools, their operation and interoperability. It covers the Petri-net frameworks (Section 4.1.1), and System Idle Time Estimation (Section 4.1.2).

4.1.1 Petri-net simulation framework

There are many Petri-net simulation and analysis tools available from a variety of universities and commercial organizations. Many of these tools are focused on the use of Petri-nets as a graphical software design notation, especially within the business work-flow arena. Others provide pure command-line analysis with limited simulation functionality. The rest lie between these extremes.

This section surveys some key tools of interest, discussing the strengths and weaknesses of each that result from the focus of each development program, with particular attention paid to the Reference-net Workshop (RENEW).

4.1.1.1 Other Petri-net Tools

DNAmaca [34] and its fore-runners were developed in the mid 1990s in the University of Cape Town (and subsequently at Imperial College in London) to model and analyze Generalized Stochastic Petri-nets (GSPNs) and their underlying Markov chains. These programs can analyze very large models, but at a price: the state space is written to disk which slows the overall computation down and the computationally intensive processing requires considerable computing power. Nonetheless, the powerful Markov chain analysis capabilities of DNAmaca are very useful, and have been exploited by a number of higher level tools, including the Imperial PEPA Compiler (ipc) [35].

A portable Java-based Petri-net tool has also been made available by Imperial College, namely the Platform Independent Petri-net Editor (PIPE) family [36]. The original version of PIPE used DNAmaca to analyze nets of any significant size, whereas the more recent version, PIPE2, performs some analysis internally. PIPE2 offers an intuitive graphical front end and will process the graphical Petri-net to present the forward, backward and combined incidence matrices, the initial marking matrix, and display the current marking and currently enabled transitions as the net is simulated. Internal state-space analysis determines whether the net is bounded, safe and live, although it does not present the reachability graph of the net.

A suite of Petri-net editing and analysis tools are presented within the “Tina” (TIme Petri-Net Analyser) [[37], [38]] toolbox, developed by the *Centre National de la Recherche Scientifique* (CNRS)’s *Laboratoire d’Analyse et d’Architecture des Systemès* (LAAS) based in Toulouse, France. The Tina suite offers a command line interface to the component tools that is very useful for integration with other tools, a feature that has been exploited in third-party applications, such as analyzing large numbers of nets as part of the fitness analysis of an evolutionary programming environment [39].

4.1.1.2 The Reference-net Workshop (RENEW)

Reference-nets are supported by the Department for Informatics of the University of Hamburg’s Reference-net Workshop (RENEW) [33], a Java-based tool that provides authoring, syntax checking and simulation tools for Reference-nets. The primary limitation of RENEW is a lack of any analytical support, but as a simulation tool (even for relatively pure and simple Petri-net models) it is very

effective. The following material gives a short introduction to the tool, a full description and tutorial [33] is available from the developers' website (http://www.renew.de/).

RENEW presents a graphical, mouse-driven user interface based around a floating toolbar as shown in Figure 4-1 below (similar in some ways to that of the aforementioned PIPE). This includes dedicated buttons for adding transitions, places, a variety of types of arc and the different forms of associated text (inscriptions, names and declarations) required for Reference-nets. The tool also supports the import and export of some non-native file formats.

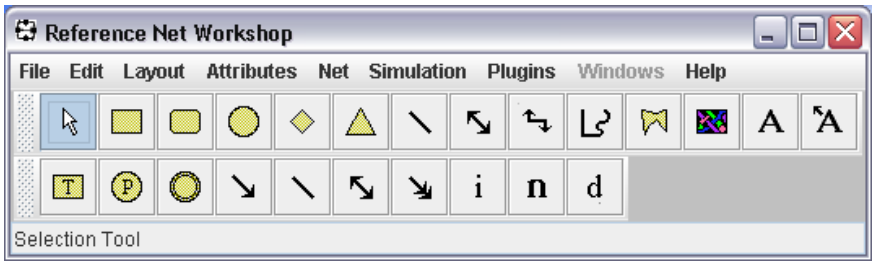

Figure 4-1: RENEW Toolbar.

The following examples illustrate some Reference-net features of interest. The colours of the entities in these example nets have no significance (from a Reference-net perspective) and are entirely user-defined, although some consistent conventions have been adopted such as formatting "import" declarations as blue and bold (see Figure 4-2).

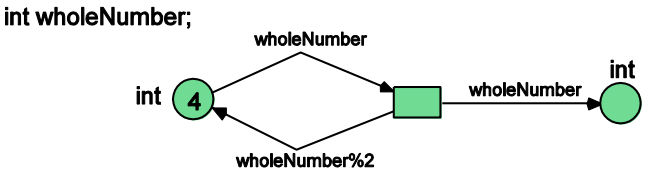

Figure 4-2: RENEW Token- and Place-typing Example.

Figure 4-2 shows the variable "wholeNumber" being declared as an integer (top left-hand corner). That variable is then used on the arcs in the net, which constrains tokens bound to that variable to be of type integer. The example also depicts places that have been declared to be of type (or color) integer, and shows the use of a Java-style inscription operator "%" (for modulus, the remainder of integer division).

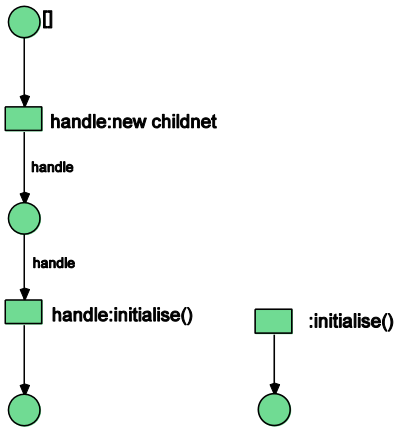

Figure 4-3: RENEW Parent and Child Net Example.

Figure 4-3 comprises two nets, the parent net on the left and the child net on the right. This shows the dynamic creation of net instances (through the constructor "new" on the first transition of the

left-hand net), the subsequent use of a token ("`handle`") as a reference to another net (namely, the net "`childnet`") and then communication between the two nets ("`handle:initialise()`").

Multiple child processes can be spawned (if there were two tokens placed in the uppermost place in the left hand, parent, net, for example, two child nets would have been instantiated).

Note that communication from child to parent is also possible, subject to the appropriate transition functions being defined at the parent level, and subject to the child being given a pointer to the parent (using keyword "`this`" to pass a reference to self to the child).

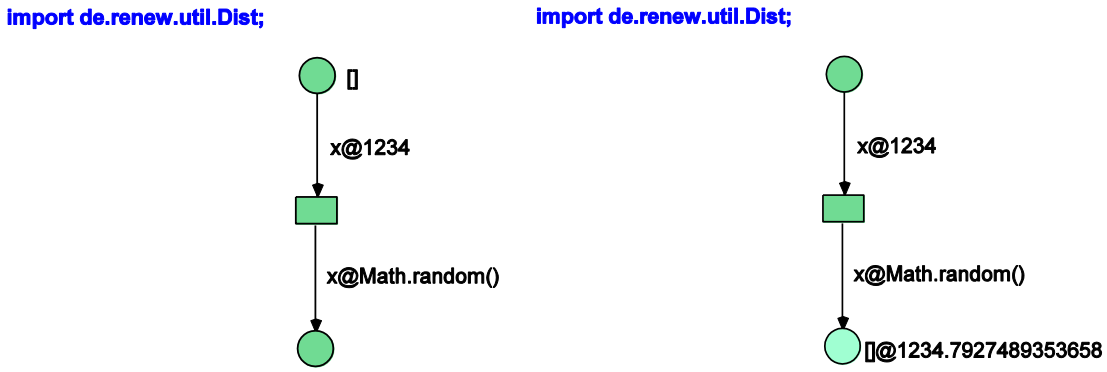

Figure 4-4: RENEW Timed Net Example.

Figure 4-4 comprises two views of the same net. On the left is the net itself; on the right a view of the net during simulation. The net itself shows two forms of timed transition: a deterministic, constant time duration (1234 time units) and a randomly generated timer duration (generated by the Math.random() method) using the "@" symbol to denote a delay. This example also illustrates the ability to import externally defined Java classes, in this case, from the de.renew.util.Dist library which includes the "Math" class.

The right-hand view of the simulation shows the end of a simulation run, with the token in the bottom place being displayed along with a time-stamp showing when the token was delivered at the place (again, using the "@" symbol).

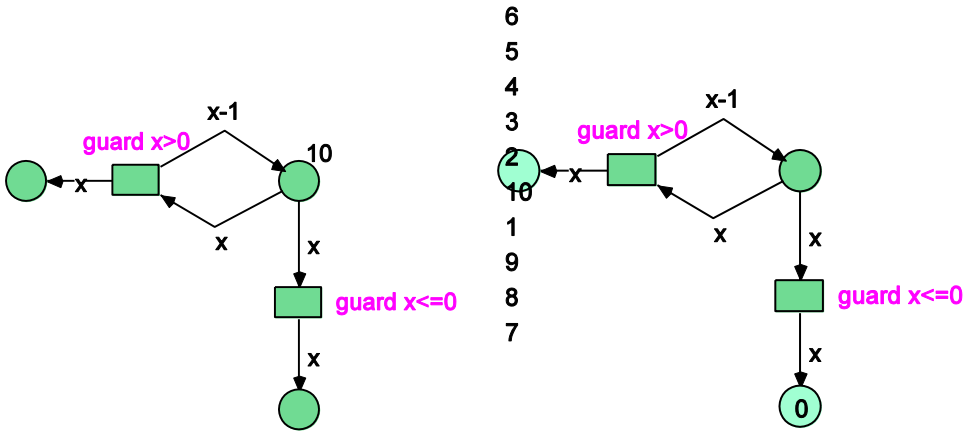

Figure 4-5: RENEW Guarded Transition Example.

Figure 4-5 also presents two figures, the example net on the left, and a view of a simulation run on the right. The transitions in the net are guarded: the transition on the left will only be enabled if the variable "x" is greater than zero, and the transition at the bottom only when "x" is less than or equal to zero. The variable "x" is initialized to ten (the initial marking can be seen in the left-hand net)

then decremented to zero, with tokens equal to its interim values deposited in the place to the left (see right-hand figure).

Many other features are also supported within RENEW, such as access to file I/O functionality via the Java inscription language, which make this a powerful simulation and modelling tool. The flipside of this, of course, is that care must be taken to keep the nets under analysis as "clean" as possible (i.e., de-cluttered with inscriptions and ancillary support code solely present to support the simulations). A useful approach to manage this exploits the modularization opportunities presented by the Reference-nets themselves—nets under analysis can be simulated as child nets in relative isolation, managed and controlled by more complex (in inscription terms) parent nets.

4.1.2 System Idle Time Estimation

An important issue in future energy-aware systems is how to estimate as accurately as possible the idle time in order to enter low energy mode and maximize the energy gains. In order to achieve this, periodic monitoring takes place to detect if the system is active or not and react accordingly. In high dense node environments similar to the ones considered in UC-02/NKUA and UC-03/NKUA [22],, traffic analysis has shown that events occur in a batch mode, meaning that an event is followed with high probability by a random number of other events as depicted in Figure 4-6.. Based on this assumption, an event-driven algorithmic solution that is based on a stochastic model has been developed that has a twofold aim; the maximization of the successful identification of events and simultaneously minimization of energy consumption. We wish to track as more events as possible (preferably all) and each node in the network to consume as less energy as possible by minimizing the sensing trials.

Figure 4-6: Events that occur in batch mode.

4.1.2.1 State of the Art Analysis

Numerous solutions have been proposed for conserving energy in wireless ad hoc (sensor) networks in literature. These protocols can be classified into roughly three (not mutually exclusive) approaches; topology control, power-aware routing, and sleep management. Topology management is an important issue because the only way to save power consumption in the communication subsystem is to completely turn off the node's radio, as the idle mode is almost as power hungry as the transmit mode. However, as soon as a node powers down its radio, it is essentially disconnected from the rest of the network topology and therefore can no longer perform packet relaying.

None of the aforementioned three approaches can optimize the energy consumption of all radio states, both active and idle, separately. Topology control and power-aware routing reduce the transmission energy of wireless nodes and do not consider the idle energy. Sleep management can reduce the idle energy by scheduling idle nodes to sleep, but does not optimize the transmission energy. We summarize the limitations of each, after providing a brief overview of the literature on the sleep management approach.

Recent studies showed that significant energy savings can be achieved by turning radios off when not in use. There are two basic approaches, namely, scheduling- and backbone-based sleep management. In the scheduling-based approach, nodes turn on their radios only in scheduled slots. The active slots of different nodes can be synchronous [1] where in a medium-access control (MAC) protocol, nodes periodically sleep to reduce energy consumption in listening to an idle channel, or asynchronous [3], such as wakeup mechanisms in ad hoc networks, which do not require global clock synchronization. Also in [4] a carrier sense media access protocol for wireless sensor networks,

employs an adaptive preamble sampling scheme to reduce duty cycle and minimize idle listening; Furthermore in [5] energy savings are achieved by powering down nodes, during idle times identified through dynamic scheduling.

In addition, several adaptive sleep schemes dynamically adjust the schedules based on traffic activities such as in [6] where a contention-based medium-access control protocol for wireless sensor networks is described, reducing the amount of energy wasted on idle listening. In [7] an extensible on-demand power management framework is proposed for ad hoc networks that adapts to traffic load. In this case, nodes maintain soft-state timers that determine transitions between the different power management modes. Finally in [8] an efficient sleep scheduling based on application timing (ESSAT) management scheme is developed, that aggressively exploits the timing semantics (traffic patterns in time) of wireless sensor network applications.

Backbone based sleep management can improve network performance by maintaining a backbone composed of a small number of active nodes, while scheduling the other nodes to operate in low duty cycles to conserve energy. In [9] a power saving technique for multi-hop ad-hoc wireless networks is proposed, where nodes make local decisions on whether to sleep or not. In [10] two algorithms are presented for routing in energy-constrained, ad-hoc, wireless networks, which turn off the radio to reduce energy consumption. Also in [11] the minimum power configuration (MPC) approach is proposed, which treats different radio states (i.e., transmission/reception/idle) in isolation.

In [12] two types of mechanisms are considered, the random sleep type where each sensor keeps an active-sleep schedule independent of another, and the coordinated sleep type where sensors coordinate with each other in reaching an active-sleep schedule. Similarly in [13], Sparse Topology and Energy Management (STEM) technique is developed, which wakes up nodes from a deep sleep state without the need for an ultra low-power radio.

In summary, existing approaches suffer from the following two major drawbacks. First, the existing approaches are only suitable for limited network conditions, as they only minimize the energy consumption under partial radio states. Power-aware routing and topology control are effective only when the network workload is so high that the transmission energy dominates the overall energy consumption of the network. Similarly, sleep management is effective only in lightly loaded networks where idle energy dominates the overall energy consumption. Second, the existing schemes may yield very different performance characteristics among different radio platforms. For example, although sleep management may considerably reduce the energy consumption when the idle power of the radios is high relative to the communication power, it is less effective for radios that have low idle power.

4.1.2.2 Simulation Models

We consider a model in which each node can operate in a certain space of states $1,K$. In each state of operation, the node scans the environment for events with a given frequency. In state-1 the frequency has its minimum value and the node idle period is the longest, while in state-K the node scans with the maximum frequency and the node idle period is the shortest.

A node can identify the existence of an event based on thresholds that are relevant to the case under consideration. For example, if the node scans its environment for increased interference, then the detection of such an event will be based on an interference level set as threshold. Among others, use cases UC-02/NKUA and UC-03/NKUA are potential candidates for the application of this methodology.

Each state i is characterized by a maximum number of failed sense trials that a node can perform before it transits to a previous state, and the frequency based on which the node scans. If a node detects the occurrence of an event, it transits to a higher state (towards state-K) in which it scans

more often. If the node does not detect an event, it transits to a lower state (towards state-1) in which sensing is sparser.

Thus, in state-1 the node consumes the least energy, but the penalty for this is the high probability of missing an event. On the other hand, in state-K the node can detect the occurrence of an event with the highest probability, but the penalty is the high energy consumption. In other words, as the node scanning trials are getting more frequent, the probability of a successful identification of an event is getting higher but the energy consumption is increased as well.

The proposed methodology tries to identify a set of parameters that will minimize the energy consumption of the node and maximize the successful detection of events. Different models for state transitions will be investigated as well. The set of parameters that are under consideration in the proposed analysis are the following:

- Number of states ,
- Maximum number of unsuccessful sense trials per state F_i. When a node reaches this number, it will transit to a lower state. Different variations of this parameter are investigated in the models we introduce,
- Granularity of the sense process per state G_i. At state i, the next scan trial of the node is based on the rule: $\text{next\_event=current\_event+G}_i$,
- We define: $G_i = \left\lfloor \frac{N_{\max}}{N_i} \right\rfloor$, where N_i is calculated linearly in the space $[1, N_{\max}]$. $N_{\max}$ is a simulation parameter and is further analysed in each of the proposed models separately.

4.1.2.2.1 Go-Back-Half (GBH) Model

In GBH, when a node detects an event, it transits from state i to the maximum state K. If the F_i criterion is met (no events were detected within the specified number of trials), then the node transits from state i to state $i/2$.

Two different alternatives are investigated in order to define when a node has to transit to a lower state:

- $F_i = N_i$: the node needs to scan N_i continuous times without detecting an event in order to transit to a lower state,
- $F_i = 1$: the node needs only one unsuccessful trial in order to transit to a lower state.

4.1.2.2.2 Go-Back-1 (GB1) Model

In GB1 model, if the node detects an event, it transits to the maximum state-K. If an event is not detected based on the F_i parameter, the node transits from its current state i to state $i-1$. Similarly to GBH model, two different alternatives are investigated in order to define when a node has to transit to a lower state:

- $F_i = N_i$: the node needs to scan N_i continuous times without detecting an event in order to transit to a lower state,
- $F_i = 1$: the node needs only one unsuccessful trial in order to transit to a lower state.

4.1.2.3 Energy Validation Based on Binary Instrumentation

In this section, we present the framework based on which we validate the energy consumption of our solution using the binary instrumentation methodology.

Program analysis tools can be categorized into two groups based on when the analysis occurs [15].

- Static analysis involves analyzing a program's source code or machine code without running it. Many tools perform static analysis, in particular compilers; examples of static analyses used by compilers include analyses for correctness, such as type checking, and analyses for optimization, which identify valid performance-improving transformations.
- Dynamic analysis involves analyzing a client program as it executes. Many tools perform dynamic analysis, for example, profilers, checkers and execution visualizers.

In software engineering, the term profiling (known also as "program profiling", "software profiling") is defined as a form of dynamic (as opposed to static) analysis of the performance of an executing program [14]. Profiling works by gathering information as the program executes under a variety of input combinations. The most common purpose of profiling is to determine which sections of a program need to be optimized so as to increase its execution speed, decrease its memory footprint, energy consumption, etc.

4.1.2.3.1 Event-based profilers and statistical profilers

Several languages (Java [15], .NET [16], Python [17], Ruby [18]) incorporate event-based profilers directly, but some profilers are based on sampling (termed *statistical* profilers). A sampling profiler periodically probes the target program's program counter using operating system interrupts. Sampling profiles are typically less numerically accurate and specific, but allow the target program to run at near full speed. The resulting metrics are not exact, but a statistical approximation, with the typical error margin of (at least) one sampling period.

In practice, sampling profilers offer a relatively accurate picture of the target program's execution, as they benefit from the fact that they are less intrusive to the actual program, and have fewer side effects (e.g. on instruction decoding pipelines and memory caches) than other approaches. Also since they don't significantly affect the execution speed, they are able to detect issues that could otherwise be masked. They are also relatively immune to over-evaluating the cost of small, frequently called routines. They can show the relative amount of time spent in user mode versus interruptible kernel mode such as system call processing.

Nevertheless, kernel code is required to handle the interrupts, which implies a loss of CPU cycles, diverted cache usage, and inability to distinguish the various tasks occurring in uninterruptible kernel code (microsecond-range activity). Dedicated hardware can alleviate these effects: some recent MIPS processors JTAG interface have a PCSAMPLE register, which samples the program counter in a truly undetectable manner. Some of the most commonly used statistical profilers are AMD CodeAnalyst, Apple Inc. Shark, gprof, Intel VTune and Parallel Amplifier (part of Intel Parallel Studio) [14].

4.1.2.3.2 Incrementing profilers

In the context of computer programming, instrumentation is defined as the ability to monitor or measure the level of a product's performance, to diagnose errors and to generate trace reports [20]. Programmers typically implement instrumentation in the form of code instructions that monitor specific components in a system (e.g., instructions may provide access to performance counters implemented in hardware). When an application contains instrumentation code, it can be managed using an appropriate management tool.

Instrumenting the program can cause changes in the performance of the program, potentially causing reduced accuracy of the results. Instrumenting will always have some impact on the program execution, typically slowing it. However, instrumentation can be very specific and be carefully controlled to have a minimal impact. The impact on a particular program depends on the placement of instrumentation points and the mechanism used to capture the trace. Hardware support for trace capture means that on some targets, instrumentation can be on just one machine instruction. The impact of instrumentation can often be deducted (i.e. eliminated by subtraction) from the results. Instrumentation is used to gather caller information and the actual timing values are obtained by statistical sampling. Instrumentation profilers can be further classified in the following categories [14]:

- Manual: Performed by the programmer, e.g. by adding instructions to explicitly calculate runtimes, simply count events or calls to measurement APIs such as the Application Response Measurement standard,
- Automatic source level: instrumentation added to the source code by an automatic tool according to an instrumentation policy,
- Compiler assisted: Example: "gcc -pg ..." for gprof, "quantify g++ ..." for Quantify,
- Binary translation: The tool adds instrumentation to a compiled binary. Example: ATOM,
- Runtime instrumentation: Directly before execution the code is instrumented. The program run is fully supervised and controlled by the tool. Examples: Pin, Valgrind,
- Runtime injection: More lightweight than runtime instrumentation. Code is modified at runtime to have jumps to helper functions. Example: DynInst,

In programming, instrumentation means the ability of an application to incorporate [20]:

- Code tracing - receiving informative messages about the execution of an application at run time,
- Debugging and (structured) exception handling - tracking down and fixing programming errors in an application under development,
- Performance counters - components that allow the tracking of the performance of the application,
- Event logs - components that allow the logging and tracking of major events in the execution of the application.

4.1.2.3.3 Simulation Framework for Energy Validation

Measuring CPU activity is telling as how busy a system is, but not how much useful work it is actually performing. A combination of metrics that include performance and energy consumption data is required for this purpose. The Intel® Energy Checker Software Developer Kit (Intel® EC SDK) provides tools to help developers design energy-efficient applications. Using the resources in the Intel EC SDK, programmers can monitor and analyze the effects their code has on the host platform to better control how the application affects energy efficiency of the host.

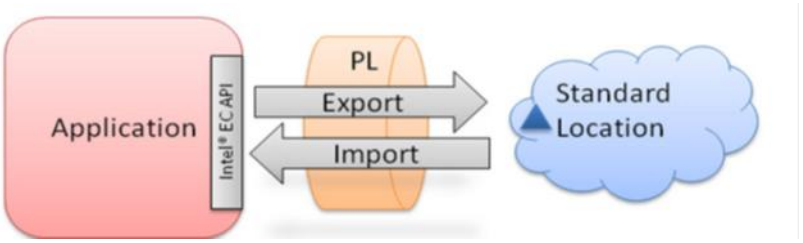

Figure 4-7: Using Intel EC to import and export counters.

The ESRV (Energy Server) (part of the Intel EC SDK) monitors a platform's energy consumption of monitored or instrumented components. ESRV provides counters for the cumulative energy consumed, as well as the immediate power draw from the power meter or power supply. The procedure for importing and exporting these counters using Intel EC is depicted in Figure 4-7. More detailed information can be found in the Intel® Energy Checker Device Driver Kit User Guide [21]. Based on the provided APIs of the ESRV we create the energy validation framework presented in the following figure:

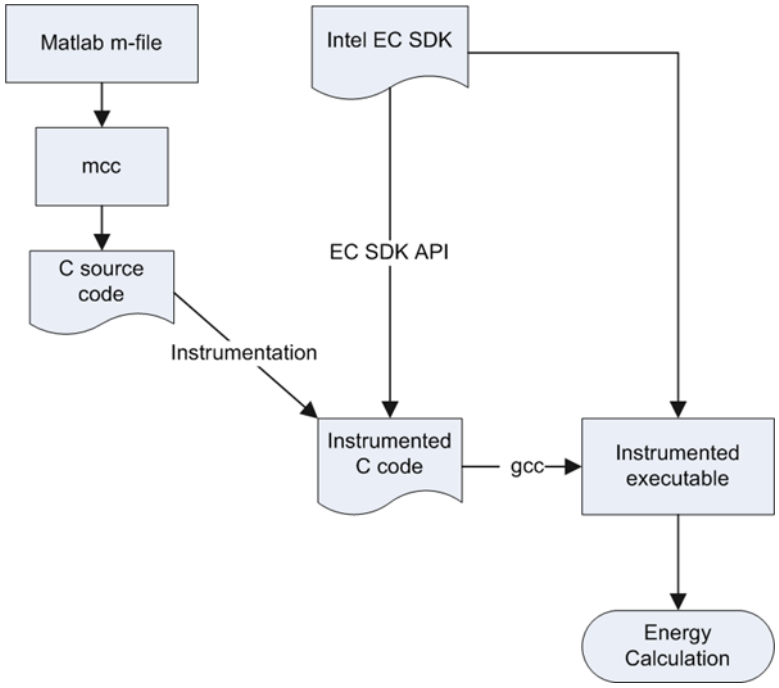

Figure 4-8: Energy validation flow.

4.2 Testbeds and Platforms

The second approach for assessing and optimizing system-level energy consumption is to use real-life experimental testbeds for implementation of proposed mechanisms, and to dynamically collect energy consumption data from active sensor nodes. A number of such testbeds are used in CONSERN, as already presented in deliverable D5.1.

4.2.1 Sensor Network Testbed – Energy Behaviour of a Sensor Node

Some first thoughts have been made in the direction of the typical operation of a wireless sensor node. Such nodes often have more or less fixed intervals for transmitting data, if they are in a normal operational mode. For instance, an air humidity sensor in an office responsible for detecting unhealthy climate conditions in a room has only to send values once a minute or within similar intervals.

If a sensor node is battery powered and is part of an energy aware network, some decision making instance in the network can make estimations about the remaining battery lifetime based on observed parameters of the node such as the current available energy of the battery, the energy costs for the periodic transmissions, and the general costs for the system itself (this should be more or less constant and much smaller than the transmission costs). In theory, such estimations could be also made by the embedded system itself, but this requires additional resources on the system and further transmission to let other nodes know the current energy state of the node.

Furthermore, if such a sensor or embedded system is part of a self-growing networking infrastructure (see also WP4 deliverables [27][28]), the battery lifetime estimations can be influenced by unforeseen events that interrupt the periodic transmissions and force the node to a higher transmission activity. For instance, the air humidity sensor could detect an unhealthy air quality condition and via an existing rule definition, a direct communication to a wireless controllable air conditioner could be made to improve the air quality including a burst mode transmission phase between both devices to recalibrate the activity of the air conditioner in a closed loop manner. After that, the air humidity sensor falls back into the normal operational mode.

Figure 4-9 shows the change of the estimated battery lifetime due to unforeseen events in a schematic way. The green line in the diagram is the original prediction of the remaining energy until the battery down event. The periodic transmission requires some energy (more than the processing on the microcontroller on the node requires) and decreases the remaining energy. However, with more or less fixed transmission intervals, a concrete battery down time can be forecasted (the related threshold is displayed as a red dashed line).

Due to an event, the node requires an additional phase of high data transmission (e.g. through self-growing or Self-X processes in the network). This is displayed as purple line that rapidly decreases the remaining energy at the beginning. After the event the node goes again to the normal operational mode and has its typical periodic transmission. This is just an example, after a reconfiguration in the network it could be also possible that the node has a new role and requires a fully different transmission interval. In this case, standard operational mode is assumed.

Figure 4-9 also shows a battery low threshold (yellow dashed line). This threshold is used as a check-point for determining whether the estimated remaining battery power is still valid or not. Due to the event, at this point of time the system must detect that the original battery down time is no longer true and the remaining power is much lesser than expected. To keep the system working until the originally foreseen point of time, a certain strategy is required. For this purpose, Figure 4-9 displays one possible solution (purple line) that simply doubles the periodic transmission interval to save some energy resources and meet the estimated battery down time.

Another approach for a recovery strategy is displayed in Figure 4-10. Here, the interval of the periodic transmissions is not changed but the transmission is reduced to the half of its original value. With this strategy, the original battery lifetime goal can be achieved as well.

It is also possible to combine both strategies or to use more complex schemes. However, each strategy is a compromise between the battery lifetime, the minimum required transmission interval, and the minimal possible transmission power. The last attribute also highly depends on the network topology or the reachability of neighbour nodes. If there are additional changes in the network or some nodes are mobile nodes, then additional recalibrations or strategy changes are required.

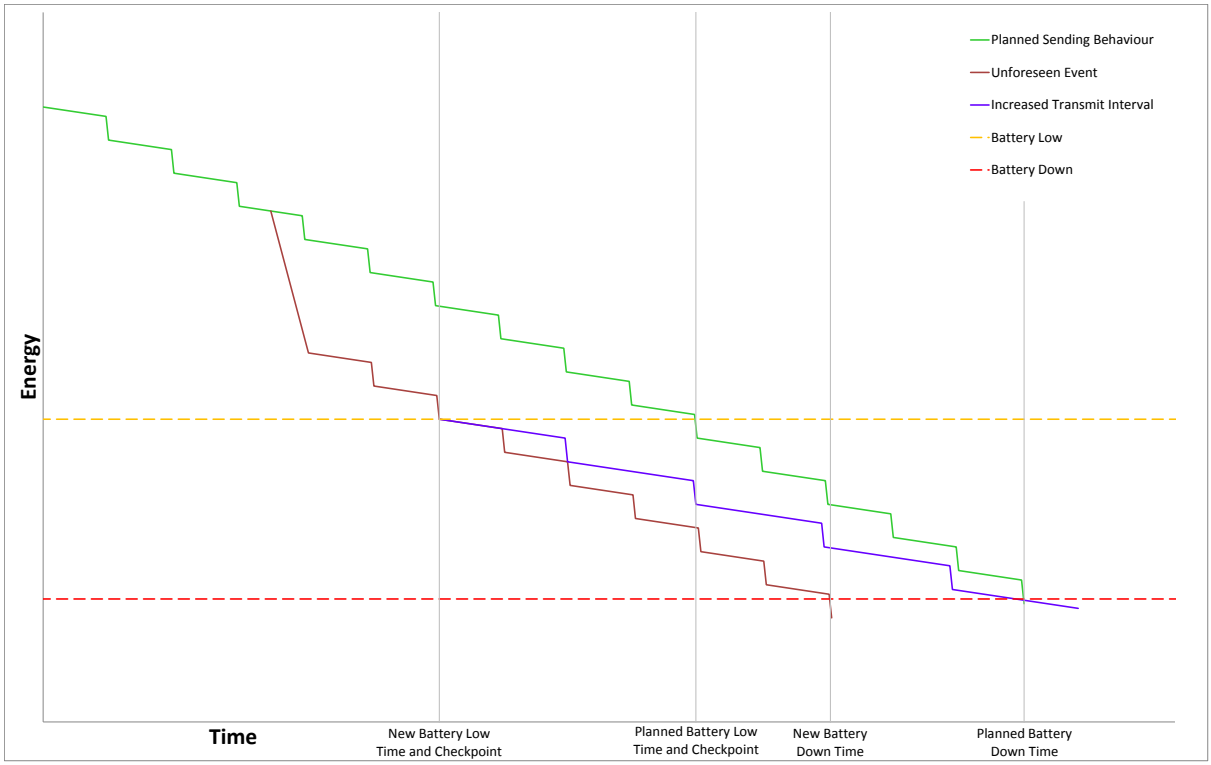

Figure 4-9: Increased transmission interval strategy to recover from an unexpected event.

The introduced concept is a very first and simplified approach. It does not consider energy consumption due to receiving data or due to protocol activities on lower layers (e.g. beacon frames send on MAC level or additional data transmissions for routing reasons on the network layer level). The next steps will be to check the outlined concept with real hardware to figure out whether the general ideas are realistic or not.

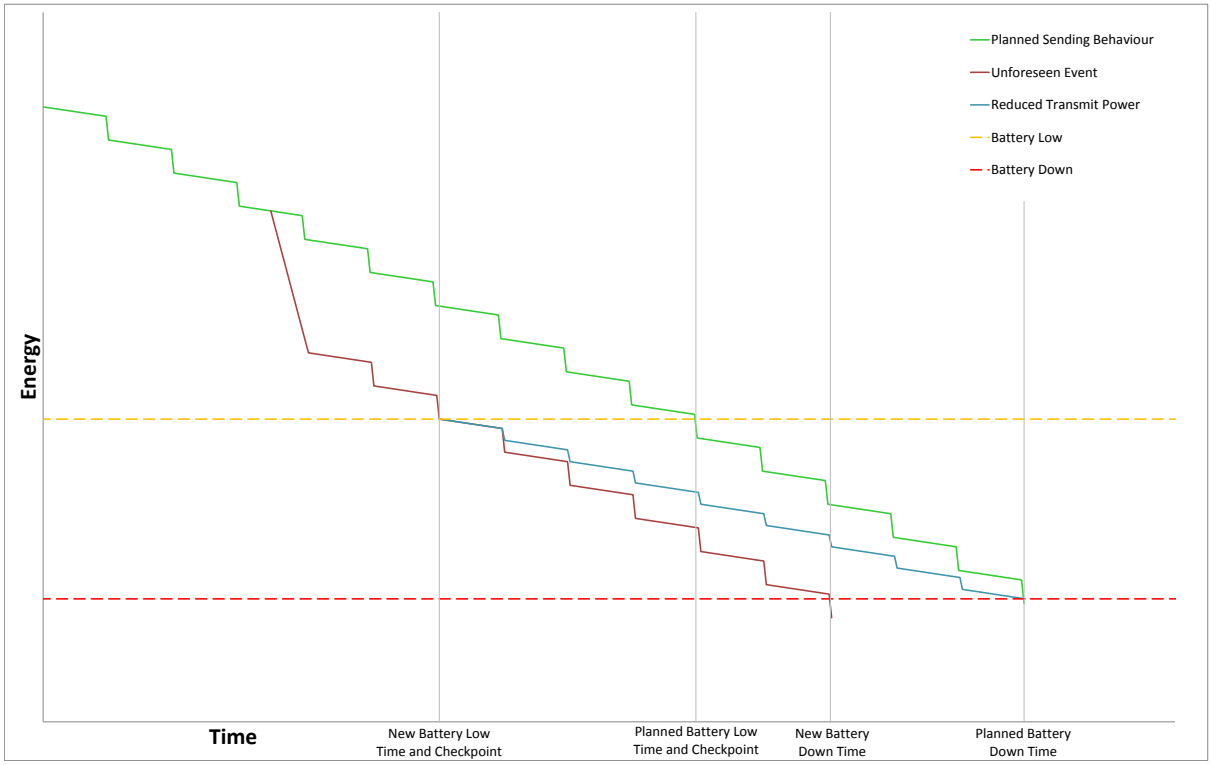

Figure 4-10: Decreased transmission power strategy to recover from an unexpected event.

Related to the testbed description for WP5 – the content will be part of deliverable D5.2, practical issues have been discussed in WP2 related to energy capabilities of the available sensor nodes. This includes some concepts for gathering data about the energy consumption of a sensor node and a short test with the battery powered XBee[1] based multi-sensor boards from Pikkerton[2] to see whether the transmitted battery voltage value decreases over the time and when the internal threshold is reached for sensing a battery low message.

To simulate the energy awareness of WSN base stations connected to IP routing devices, it could be also possible to use an energy metering power plug (e.g. ZigBee device from Pikkerton) together with the routers to measure the energy consumption indirectly via a secondary device. The power plug monitors the consumption of the router and a software daemon on the router subtracts the consumed energy from a "virtual" battery. The status of the virtual battery can be provided to the decision making backend. The power plugs that are attached to the routers could be used by the backend to switch on additional routers when they are needed or to switch them off respectively. This would be ideal in emergency situations when the system needs better coverage (802.15.4/ZigBee or Wi-Fi) due to the current situation.

By switching on additional routers, we can simulate the process of reconfiguring a sensor node to be temporarily a Wi-Fi access point, because real sensor nodes which could be reconfigured to have Wi-Fi on-board (simultaneously with other technologies like 802.15.4/ZigBee) are not available in the current testbeds (except of the base station nodes that are a different class of devices). In this case, the combination of a ZigBee node and a connected Wi-Fi access point can be seen as a single black box from the perspective of the decision making system. Theoretically, the ZigBee node could also use the ZigBee base station that could be inside the new activated router to connect to a different PAN and benefit from the new available coverage of the extended Wi-Fi overlay network (that also interconnects WSNs). However, the required reconfiguration overhead is really huge and there is a good chance to finally lose the connection to the ZigBee power plug, if the new PAN is not available or the Wi-Fi access point is not able to connect to the infrastructure.

The sensor nodes from Pikkerton (power plugs and multi sensors) use XBee modules to process the ZigBee communication. The modules require 115mW when transmitting, 125mW when receiving, and 50mW when they are idle. The XBee module could be configured by software to switch into a sleep mode and wake up at specific intervals for transmission to conserve power. It is also possible to reduce the transmission power of the signals, but this has only effects on the power consumption, if the module sends messages. Receiving messages requires always the same power and is not affected by the current transmit power settings. However, the cycle for listening on the radio can be changed to reduce the energy consumption a bit. The configuration of the XBee is controlled via AT commands, which are either sent plain via an RS232 connection or especially encapsulated via the RF interface to configure remote nodes.

[1] 802.15.4/ZigBee RF modules from Digi (see http://www.digi.com/products/wireless-wired-embedded-solutions/)
[2] German company that produces ZigBee based sensors and power plugs (see http://www.pikkerton.de/zigbee/ZigBee.html)

5. Results

This section reports on results and measurements made using the three main approaches of this work. The benefits of the centralization/decentralization mechanism as simulated by the Petri-net approach are given. The tradeoffs in power state selection strategies are illustrated. A performance evaluation for Non-Intrusive Aggregation is performed, and sensor battery discharge is explored in a ZigBee senor network.

5.1 Petri-net Energy Modeling

The system simulator is set up to output the number of a set of different operations within the network:

- Number of sensor samples, ***num_samp***. We neglect the power requirement of sensor reading operations in these system simulations since they will be constant across the different systems. See D2.2 for a treatment of what occurs in the sensor itself.
- Number of transmission to the gateway, ***global_trans***.
- Number of local transmissions, ***local_trans.*** These are only between nodes only in the decentralized network.

Figure 5-1 shows how the number of local transmission and transmissions to the gateway scales with the size of the network, both for centralized and decentralized modes. The number of transmissions is scaled per sensor sample to make fair comparison between the different modes, different network sizes and simulation lengths.

For the centralized mode the number of gateway transmissions per sample are constant across the network size, as expected, and the value given in the simulations (0.33) is what is expected since the sensor is running in a mode where it reads three samples for every summary it sends across the network. This is a good check that the simulation is operating correctly in this mode. In this mode all transmissions are direct to the central gateway node.

For decentralized mode we see that as the network grows in size the number of global transmissions per sample drops, with the number of local transmissions increasing to compensate. For trivial networks (of a couple of nodes) decentralized and centralized are identical in layout.

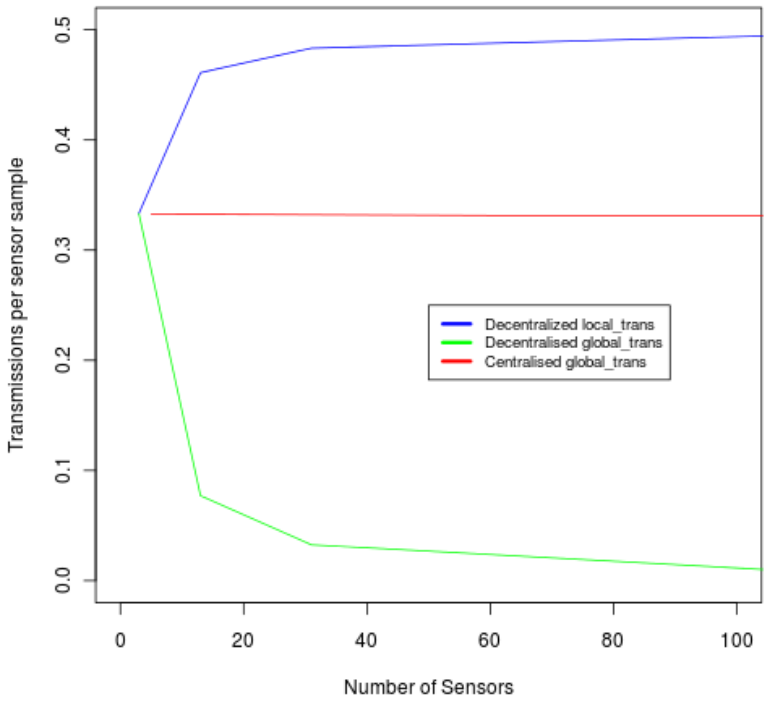

Figure 5-1: Number of transmissions in centralized and decentralized modes of operation.

In order to explore the tradeoffs that decentralized mode makes in terms of power consumption, we need to know the relative difference in power cost between global and local transmissions. Since this varies due to node placement and environment conditions, we assume an average ratio across the whole network and see how this affects the power consumption of the decentralized versus centralized modes. Figure 5-2 shows the energy overhead for a range of ratios from 1.6 to 3.0. Parts of the curves above the red dotted line are where the decentralized mode has a higher energy cost than centralization, and below those where decentralization has energy benefit. The 1.6 to 3.0 range is chosen for the plot since below 1.6 there is never any benefit from decentralizing, and above 3.0 there is benefit from decentralization even for small networks of fewer than 10 nodes.

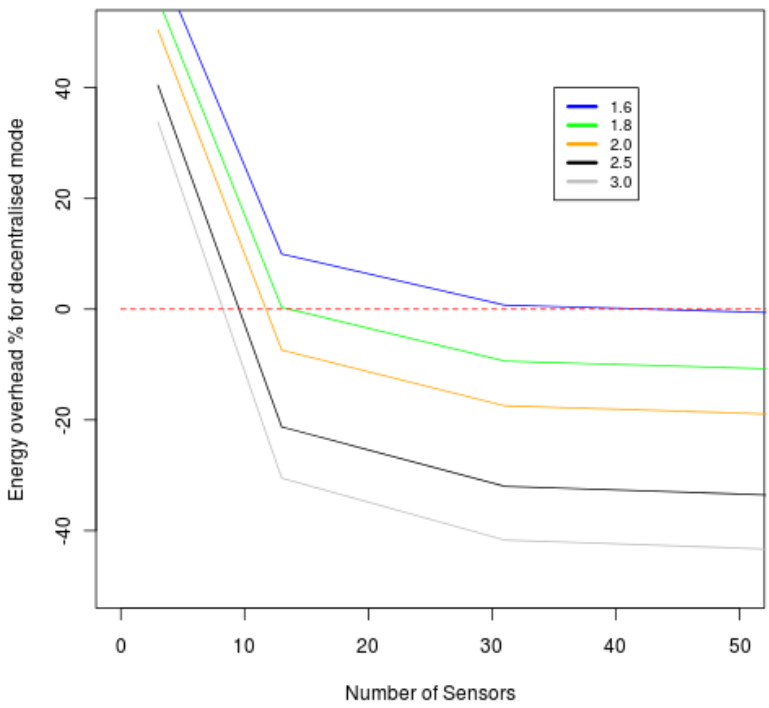

Figure 5-2: Comparison of energy overheads for decentralized modes for a range of different power factors.

Between these extremes there is a network size range (from 10 to 20 nodes) where the network can benefit from a switch to decentralization. Thus an adaptive energy aware network should switch between centralized and decentralized modes as the network grows across these boundaries.

Future work can explore in detail when this switch can happen, but this work gives some bounds on network sizes and transmission levels over which this technique can be useful. More detailed simulations with a more accurate transmission power model can be built and algorithms that detect when to switch can be developed.

5.2 System Idle Time Estimation

5.2.1 Metrics

For the evaluation of our approach, two metrics are introduced:

- Successful Detection Percentage (SDP): the percentage of successfully detected events, and,
- Energy Consumption Percentage (ECP): the percentage of the energy spent to identify events.

In order to select the optimum points, we introduce the success energy product (SEP), which is given by the following equation:

$$SEP = SDP \times (1 - ECP)$$

The optimum point is the point that maximizes SEP. Notice that by adding weights in any of the two factors of the product, optimum points that favor this factor are generated.

The proposed metrics are calculated based on the scenario in which the node keeps sensing for events constantly with the higher possible rate. In this case, all the events are identified but the energy consumed is the maximum.

5.2.2 Go-Back-Half (GBH) Model

As presented in section 4.1.2.2.1, in GBH, when a node detects an event, it transits from state i to the maximum state K. If the F_i criterion is met (no events were detected within the specified number of trials), then the node transits from state i to state $i/2$.

Two different alternatives are investigated in order to define when a node has to transit to a lower state:

- $F_i = N_i$: the node needs to scan N_i continuous times without detecting an event in order to transit to a lower state,
- $F_i = 1$: the node needs only one unsuccessful trial in order to transit to a lower state.

In Figure 5-3, the SDP and the ECP are presented when $N_{\max} = K$. For example, when $K = 10$ then, $N_{\max} = N_{10} = 10$. The distribution of N_i is linear in the space of the K states. In Figure 5-3, it is obvious that the model provides better ECP for $F_i = 1$, but worse SDP than $F_i = N_i$ for $K > 5$. Also, the highest SDP is achieved in both cases when $K = 2$ and is equal to 98% and corresponds to an ECP is equal to 64%. However, with $K = 8$, SDP is equal to 87% and the ECP is equal to 39% when $F_i = 1$, which provides a very good SDP with great reduction in ECP.

The SDP curve for $F_i = N_i$ is decreasing when $K \in [2,15]$ because the node remains in low states and thus it misses to successfully detect any events. This is also the reason that the ECP following a decreasing trend as well. But from state-15 and above, the node remains more time in higher states and thus both the SDP and the ECP are getting increased. On the other hand, when $F_i = 1$, as ECP is always decreasing as the number of states increase since more states provide better granularity and at the same time do not affect the time the node needs to transit to a lower state as the F_i is independent of the N_i. However, the increase in the number of states affects negatively the SDP because the node spends more time in lower states and thus misses more events.

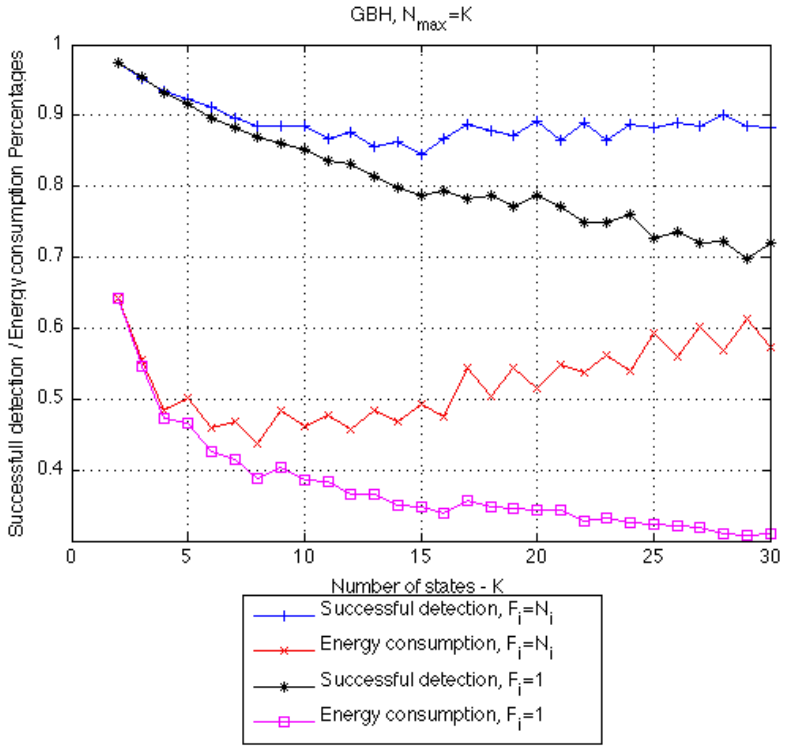

Figure 5-3: GBH model – The maximum number of measurements at state K is equal to K.

In Figure 5-4, the behaviour of GBH is depicted for $N_{\max} = 3K$. In each state, the maximum number of measurements is three times higher than the previous case. In this case, when $F_i = N_i$, as the number of states increases, both the SDP and ECP are increased. This is expected since by increasing the N_i, we achieve higher G_i and more events can be detected. However, since $F_i = N_i$, a node stays longer in a high state compared to the first case and thus the ECP is increased as well. The maximum SDP is reached when $K = 30$ and is equal to 98%. The respective ECP is equal to 81%.

On the other hand, when $F_i = 1$, both the SDP and the ECP decrease as the number of states increases resulting to an ECP equal to 22% for an SDP equal to 52% for $K = 30$. From these results, we can extract that when $F_i = 1$ and the $N_{\max}$ increases, the model gives better ECP but with the price of low SDP. On the other hand, when $F_i = N_i$ the model behaves differently since the SDP increases but with the price of an increasing ECP.

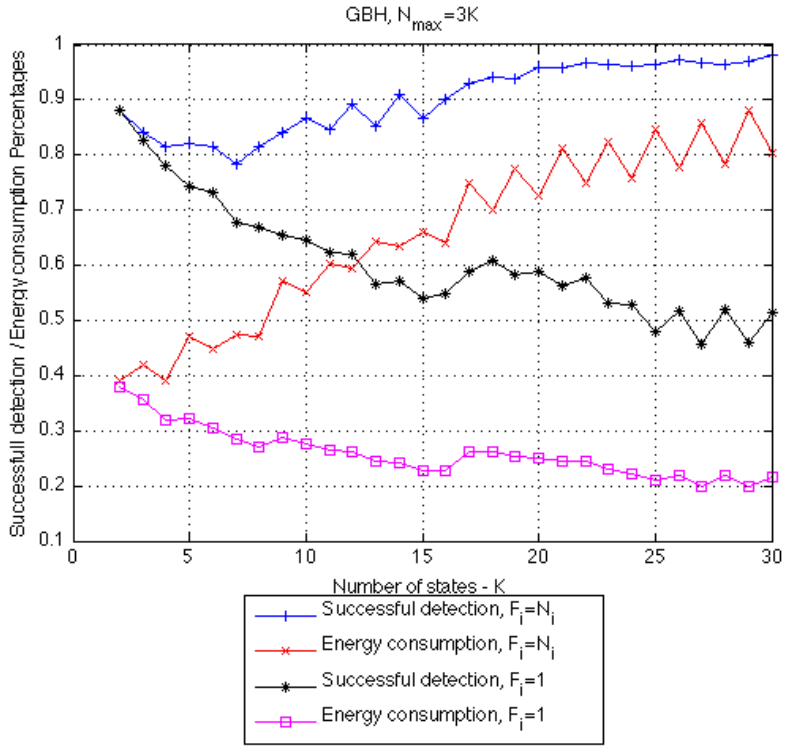

Figure 5-4: GBH model – The maximum number of measurements at state K is equal to 3K.

In Figure 5-5, we consider $N_{\max} = 6K$. As depicted in the figure, the behaviour of the model is similar to the previous case where $N_{\max} = 3K$. However, for $F_i = N_i$ they are both shifted higher yielding better SDP but with higher ECP. On the other hand, when $F_i = 1$, the model behaves as in the previous case where $N_{\max} = 3K$ but with the curves shifted lower. The low ECP achieved in this case is expected because this model can very fast transit to a very low state since only one failed measurement per state is enough to transit lower. However, as the number of states increases, G_i for the lower states is very high and a lot of events are not detected resulting in very low SDP as well.

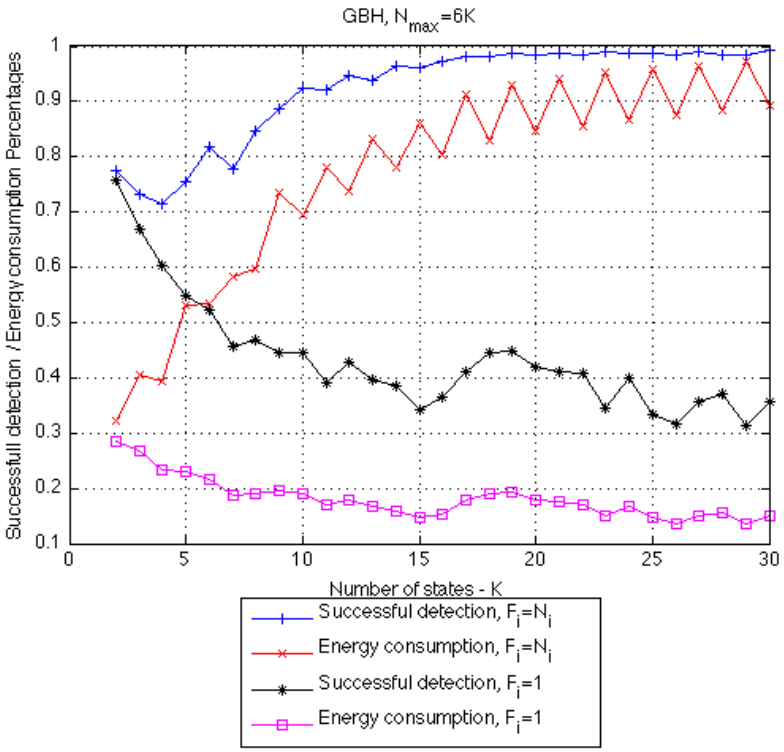

Figure 5-5: GBH model – The maximum number of measurements at state K is equal to 6K.

5.2.3 Go-Back-1 (GB1) Model

As described in section 4.1.2.2.2, in GB1 model, if the node detects an event, it transits to the maximum state-K. If an event is not detected based on the F_i parameter, the node transits from its current state i to state $i-1$. Similarly to GBH model, two different alternatives are investigated in order to define when a node has to transit to a lower state:

- $F_i = N_i$
- $F_i = 1$

In Figure 5-6, the behavior of GB1 is depicted for $N_{\max} = K$. For $F_i = N_i$, the best trade-off between SDP and ECP is achieved for $K = 4$ where SDP is equal to 95% and ECP is equal to 55%. For $F_i = 1$, the model behaves better since for $K = 4$, SDP is equal to 94% but with a lower ECP equal to 51%. As the number of states increases, both SDP and ECP are increased for both cases. When $F_i = N_i$ and $K > 20$, the node does not manage to transit to lower states and thus the SDP and ECP reach their maximum values. When $F_i = 1$, the SDP remains in high levels around 96% while the ECP remains under 80%.

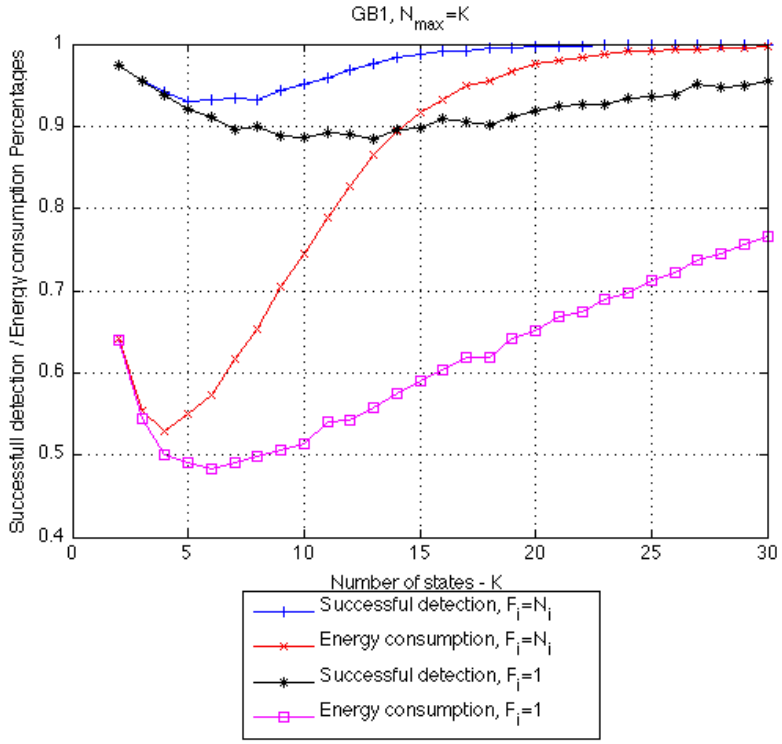

Figure 5-6: GB1 model – The maximum number of measurements at state K is equal to K.

For $N_{\max} = 3K$ and $F_i = N_i$, as the number of states increases, the SDP and ECP reach their maximum values very fast ($K = 12$). On the other hand, when $F_i = 1$ the model behaves differently. Both SDP and ECP increase as the number of states increases, but not as radically as when $F_i = N_i$. The reason is that this model can reach lower states faster and thus spent less energy.

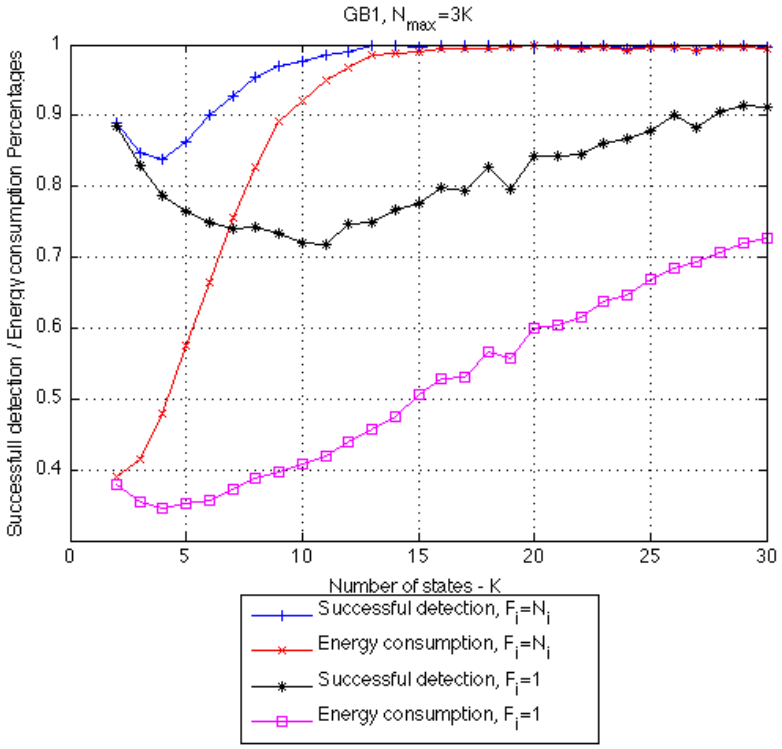

Figure 5-7: GB1 model – The maximum number of measurements at state K is equal to 3K.

For $N_{\max} = 6K$ and $F_i = N_i$, GB1 follows the same behavior as before with the difference that the SDP and the ECP reach their maximum for a lower number of states ($K = 9$). On the other hand, when $F_i = 1$ and the number of states is low, both SDP and ECP are lower compared to the case where $N_{\max} = 3K$. However, as the number of states increases, they reach the same values as before with the SDP equal to 95% and the ECP equal to 78% for $K = 30$.

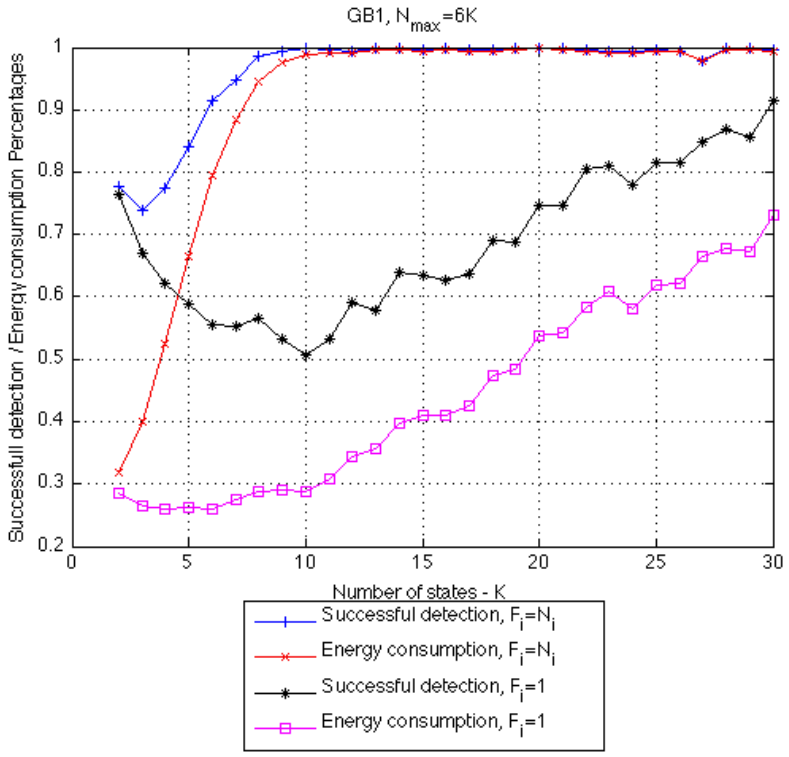

Figure 5-8: GB1 model – The maximum number of measurements at state K is equal to 6K.

5.2.4 Summary of optimum points

In the following table, we summarize the optimum points (optimum number of states) per scenario. The optimum points were selected based on the SEP metric when both the SDP and ECP have the same weight. For each scenario investigated in our analysis, the SEP values were calculated and the highest ones were selected. The table presents the optimum number of states per model and per $N_{\max}$ value. From the results, It is obvious that for the majority of the scenarios the optimum number of states is equal to two. Exception to this is the case where $N_{\max} = K$ for which both models have optimum points for larger number of states.

Table 5-1: Summary of optimum points

	GBH		**GB1**	
	$F_i=N_i$	$F_i=1$	$F_i=N_i$	$F_i=1$
$N_{max}=K$	8	8	4	6
$N_{max}=3K$	2	2	2	2

N_{max}=6K	2	2	2	2

Notice that the optimum points presented in this analysis are based on a certain metric (SEP) selected for presentation purposes. If another metric is selected, optimum points may vary. Thus, the proposed approach provides a methodology and a set of results that show the potential system architect how to select the optimum values for the nodes operation based on specific requirements. For example, if we consider a requirement restricting all the nodes that operate based on the proposed model to match a minimum SDP equal to 90%, the optimum points would be different.

5.2.5 Energy Validation Based on Binary Instrumentation

Using the flow presented in Figure 4-8, we are able to validate the energy consumption of the proposed schemes versus the trivial scheme in which the node keeps sensing for events constantly with the higher possible rate. A screenshot of the profiling process is presented in Figure 5-9 and the related validation data are provided in Figure 5-10.

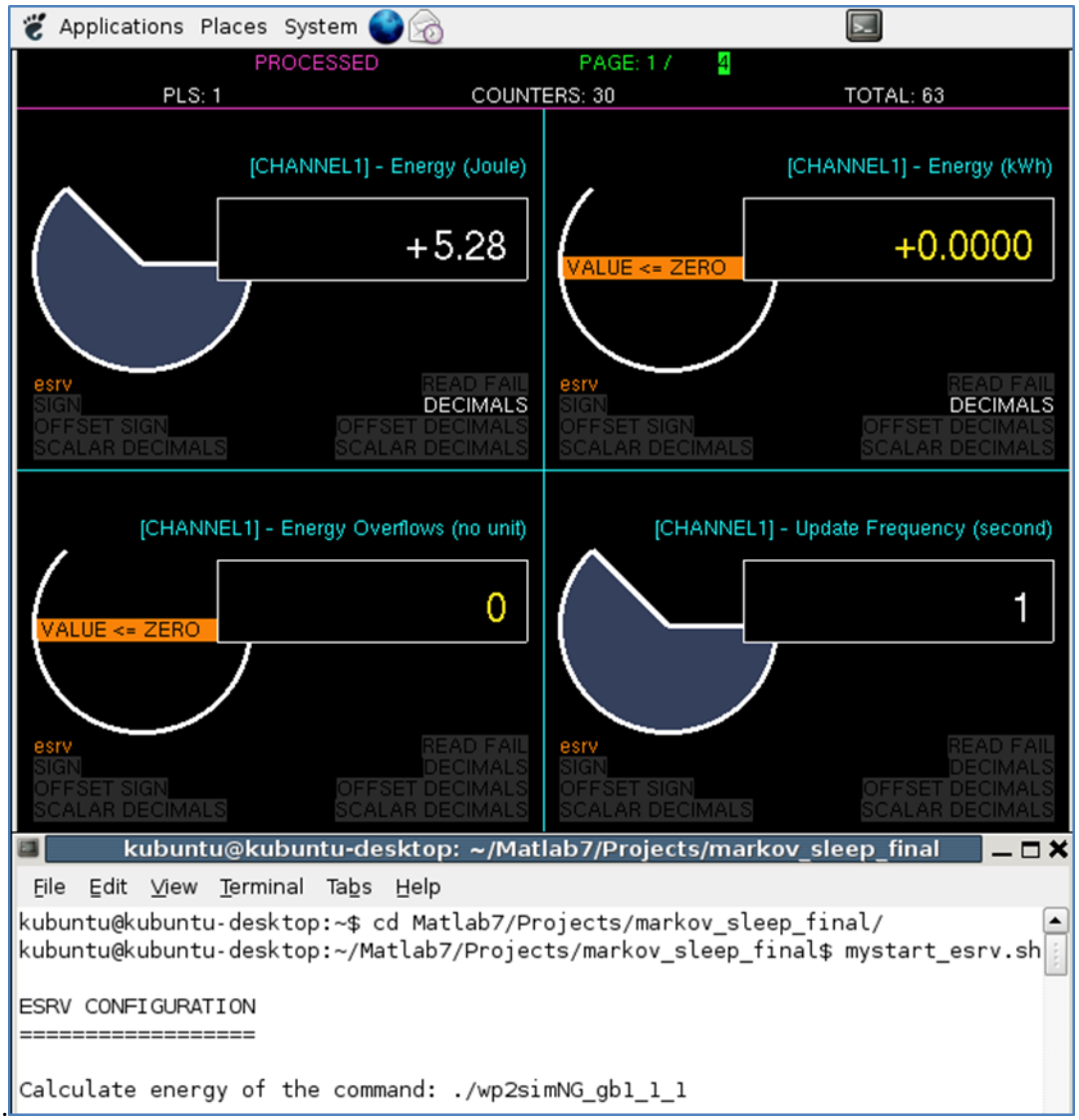

Figure 5-9: Profiling screenshot.

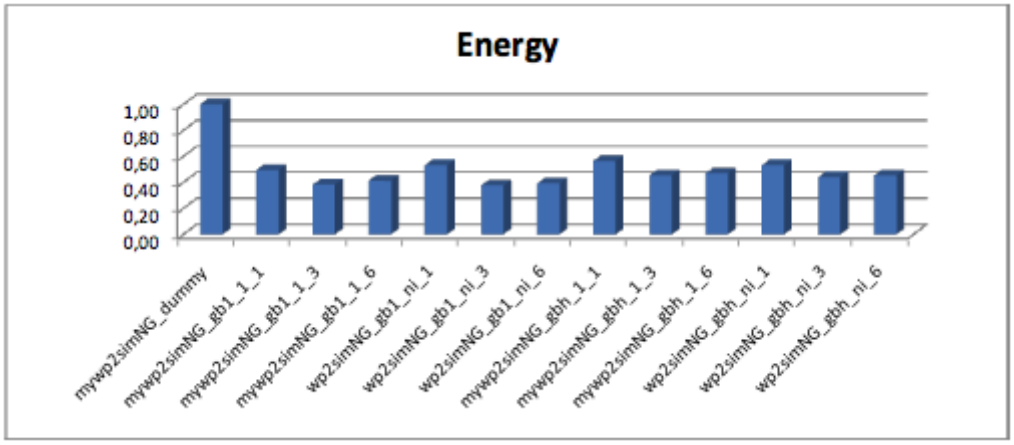

Figure 5-10: Energy validation data.

In the above figure, the name for each bar is of the form "mywp2simNG_model_Fi_Nmax". For example, mywp2simNG_gb1_1_3 describes the model GB1 for the case where $F_1 = 1$ and $N_{\max} = 3K$.

Note that only relative energies are reported because the platform where the software is validated is Intel x86, therefore absolute values would have little relevance with regard to actual networking elements executing the same algorithms. Nevertheless, relative results show that the energy for executing the proposed schemes is significantly less compared to that of the trivial case; therefore the energy gains derived from the "intelligent" derivation of the sleep time will be quite significant.

5.3 Performance Evaluation of Non-Intrusive Aggregation

To evaluate the performance of the Non-Intrusive Aggregation, we performed real-life experiments on the IBBT W-iLab.t wireless sensor testbed [31]. The W-iLab.t testbed consists of about 200 TmoteSky sensor nodes, spread out over three floors. Some of the sensor nodes are located in ventilation shafts between the floors. As such, these ventilation shafts provide a connection corridor through which communication is possible between different floor levels.

For our evaluation, the number of information types on each node was limited to two types of data traffic and two types of control information between direct neighbours. A larger number of information types would have given an overly optimistic perspective since global aggregation is able to combine information exchanges coming from different network layers. Each node has two separate applications that send measured information to one sink. In addition, networking protocols send notification messages to neighbouring nodes. These information exchanges contain status information, such as the remaining energy or the signal strength. This information is typically used by neighbour discovery, slot assignments, link quality estimations and synchronization protocols.

Five different aggregation mechanisms were compared.

- **No aggregation:** This is used the reference scenario.
- **Packet combination:** This is the most commonly used aggregation mechanism for Wi-Fi networks. Created packets are delayed for a short time at the MAC or PHY layer before they are sent over the wireless network. If multiple packets to the same next hop address are delayed in this way, they are combined in a single MAC or PHY frame.
- **Traditional data-aggregation:** This is the most common non-intrusive aggregation mechanism for wireless sensor networks. Each application generates information which is

encapsulated in a packet. In intermediate hops, the packet is decapsulated and the payload is offered to the application. The application can choose to fuse its own measured data with the received information. It then forwards the packet on towards its destination. Data packets coming from different applications are not aggregated together.

- **Joint-application data-aggregation:** With this approach all applications are jointly designed. The result is similar to traditional data-aggregation, but the resulting application can combine data packets from all original application. In our experiments the joint-design includes the two application data streams, but the control packets are not aggregated.
- **Global aggregation:** This is our non-intrusive Aggregation. Applications can be maintained by different developers, but their data is combined as if using joint-application data-aggregation. In addition, control packets are also aggregated, without the need for introducing any dependencies between the network layers.

In one set of experiments, we explored the dependence of the benefit of aggregation on the size of the network. In this scenario the intervals between data and control packets are fixed (60 sec.), the tolerance to packet delay is fixed (30 sec.), and the network size is varied from half a floor (40 nodes) to three floors (about 200 nodes). The resulting average packet transmissions per minute per node are shown in Figure 5-11.

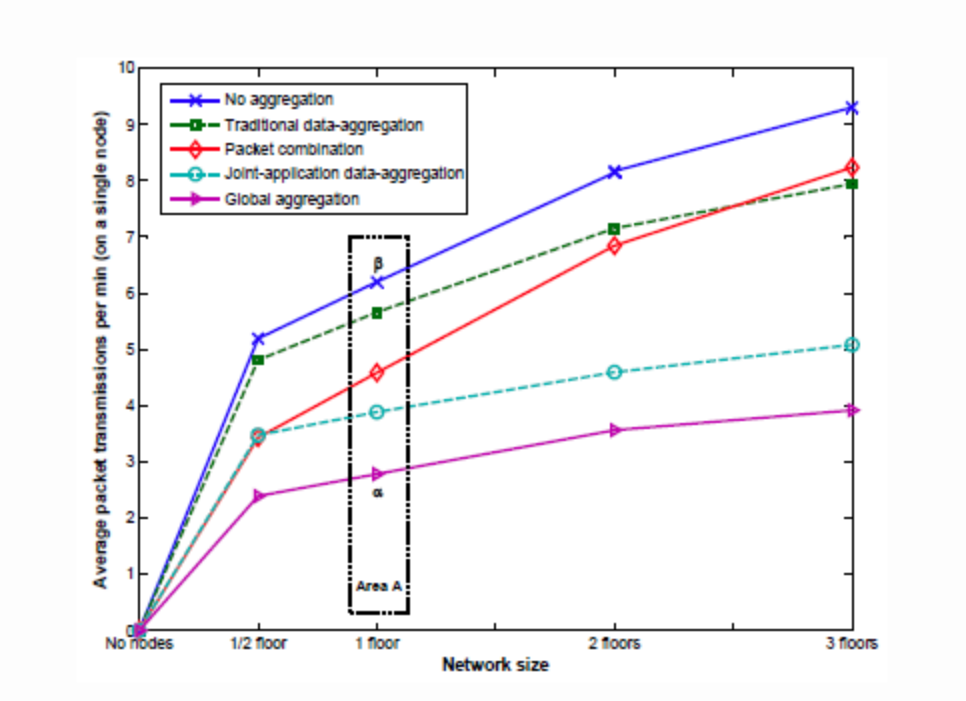

Figure 5-11: Aggregation benefit as funtion of Network Size.

Depending on the type of application, the time interval between sensor measurements can vary significantly, from a few seconds for fire-detection applications to several hours for the monitoring of long-term nature phenomena. Similarly, unreliable networks with mobile nodes or frequently occurring node failures require more frequent status updates than static sensor networks. Since existing aggregation protocols are often optimized for data traffic, the ratio of data to control traffic may have a substantial influence on the performance of aggregation. Therefore, in a second set of experiments the size of the network is kept constant (one floor, approximately 80 nodes), as well as the intervals between control packets (60 sec.), but the intervals between data packets are varied. The resulting average packet transmissions per minute per node are shown in Figure 5-12.

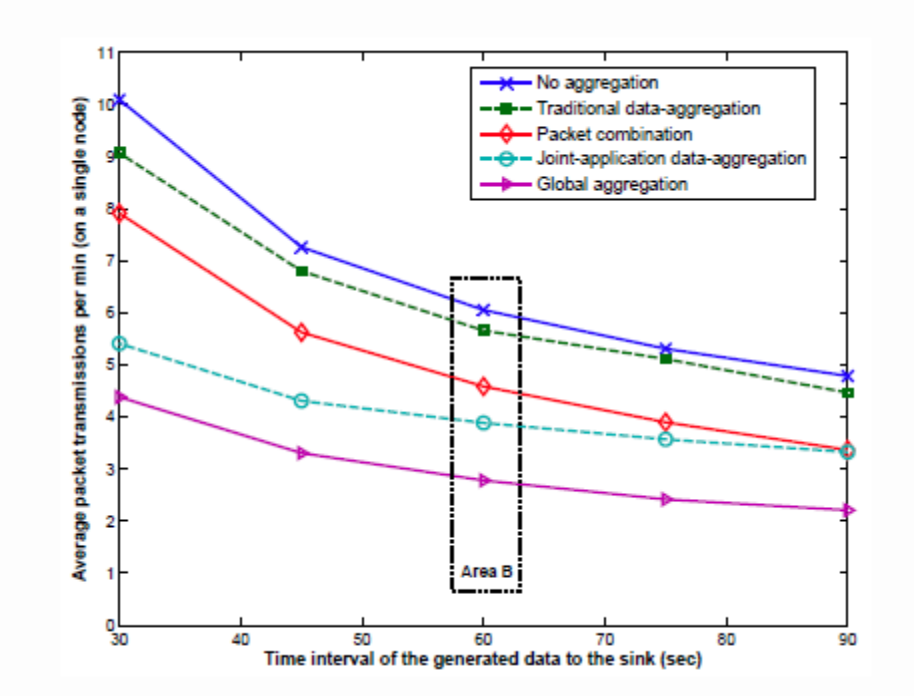

Figure 5-12: Aggregation benefit as function of the ratio between data and control traffic.

To put the two diagrams in perspective, Area B of Figure 5-12, in which the interval between data packets is 60 sec, corresponds to Area A of Figure 5-11, in which the size of the network is one floor.

5.3.1 Power Saving

As mentioned earlier, power saving is achieved mainly by putting the radio to sleep mode. By intelligently turning the radio on and off, the amount of time spent in low-power sleep mode can be increased. However, the sleep time is limited by the total throughput that the MAC protocol needs to support. Thus, if fewer packets need to be sent, the sleep duration can be increased without causing network congestion. As an example, Table 5-2 shows the expected energy consumption (estimated from [32]), associated with different average throughputs for the popular S-MAC and B-MAC protocols. In the previous sections it was demonstrated that our Non-Intrusive Aggregation can reduce the number of packet transmissions (and resulting average throughput) by more than 50%. We can conclude from Table 5-2 that this reduction in turn decreases energy consumption by 30-50%.

Table 5-2: Estimated power consumption with the S-MAC and B-MAC protocols.

Average Throughput (bps)	S-MAC average power (mW)	B-MAC average power (mW)
25	4	5
50	7.5	7
75	11.5	9
100	15	11
150	23	13
200	31	16

5.4 Measurements from a real ZigBee Sensor Network

To measure the actual energy consumption of the Pikkerton ZigBee multi sensors, a simple test scenario setup has been established containing an XBee coordinator connected to a PC. The PC is running observation software that actively polls the multi sensor to get the current battery power and save the data in a log file. This polling can be configured to a higher interval to force a battery down and collect the battery measurements until this event. These measurements can be used to identify the discharge behaviour of the battery and predict the average life time of a node.

The measurements results show that the general theoretical approach is valid, but too naive (see Figure 5-13 and Figure 5-14). Both figures illustrate different measurements of a real ZigBee multi sensor over a twelve hour time period. Figure 5-13 shows the measured battery voltage values and the battery status (OK or LOW) including time frames without receiving any packet from the node. After a longer non-active time period the battery recovers a little bit. However, each transmission has its energy costs and the diagram shows that there is a good chance to get a battery LOW status below approximately 3.5V. The values are also fluctuating a bit, but the general trend is clearly visible. The curve follows a typical battery discharge characteristics that can be used to predict the point of time when the battery goes into a critical battery state based on transmission intervals and power. This also means that the initial theoretical approach using linear characteristics is too optimistic in its prediction.

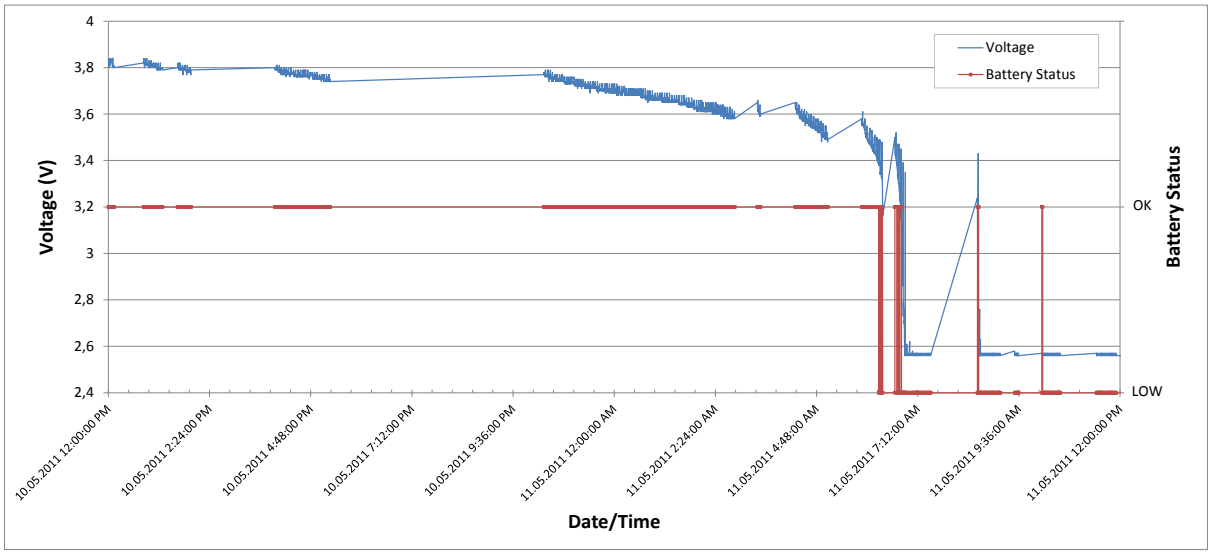

Figure 5-13: Change of battery voltage and status of a ZigBee sensor with real time intervals.

Figure 5-14 shows the values again compared with the actual packet numbers or the packets that are send during the measurements not using correct time framing and ignoring intervals without sending packets. The general characteristic is the same, but it shows how much packets can be still send without running in a critical state. This can be also used for predictions that are based on the possible capacity of a node to send further packets with a given signal strength and a more or less fixed packet size.

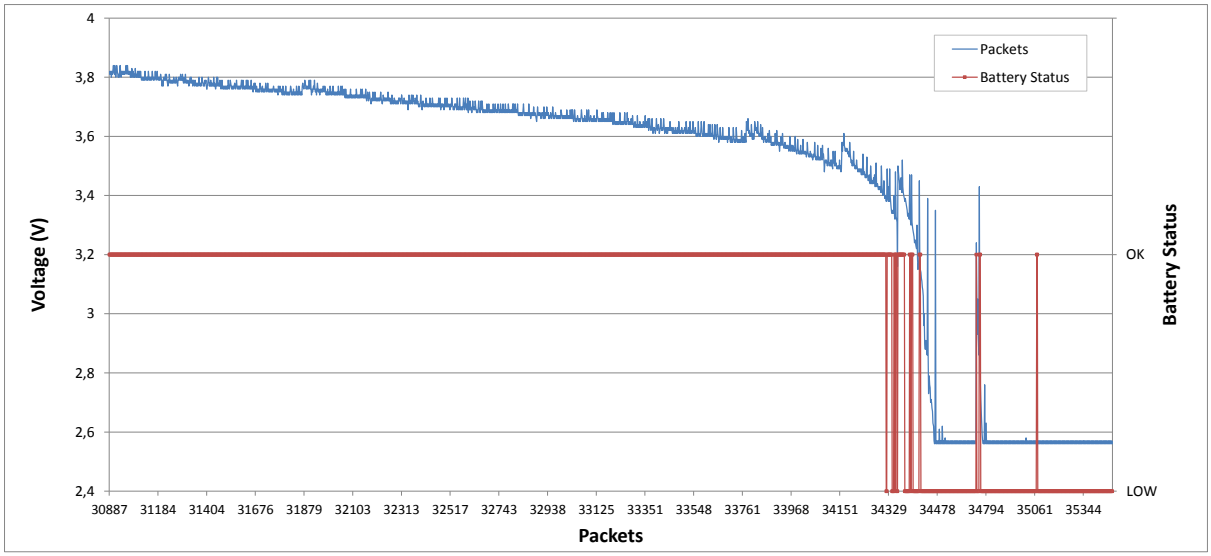

Figure 5-14: Change of battery voltage and status of a ZigBee sensor compared with packets sent.

The battery status is also fluctuating after the initial threshold for a battery low event. This is maybe due to some recovery phases after longer non-active periods and can be ignored. Both, curves the time based and the packet based could be used for battery life time estimations, also by combining them. However, the curves are node specific and dependant on several parameters like signal strength, transmission intervals, internal traffic overhead in the ZigBee networking layer (e.g. after reorganisation or re-joining a PAN), or other unexpected events in the network. The next steps in WP2 should be identifying the concrete parameters, validating the different battery behaviours, finding maybe a general formula, and giving rule sets to WP4 for the cognitive elements to enable energy awareness as one scenario of a general self-growing network.

6. Conclusion

This deliverable has looked at a variety of energy optimisation mechanisms from the perspective of whole network systems. The main findings of the simulation/testbed work on system level energy consumption are:

The Petri-net simulation technique allows models of centralised and decentralised sensor polling applications to be developed. For sensible values of global and local transmission power levels, there is benefit from a switch from centralised to decentralised mode at networks of size around ten to twenty nodes, sizes which are appropriate for the home networking scenario that has been targeted.

The System Idle Time estimation approach looked at two strategies of power level selection, and found trade-off points based on different system parameters (such number of power states), and a desired system property metric based on successful event detection rates and event energy consumption percentages. The results from the binary instrumentation tool show that the energy for executing the proposed schemes is significantly lower; therefore the energy gains derived from the "intelligent" derivation of the sleep time has the potential to be quite significant.

Real-life testbed implementation of the Non-Intrusive Aggregation algorithm demonstrated how the number of packet transmissions could be dropped using a range of different levels of aggregation. In turn this reduction in packet rate is interpreted in terms of energy reduction in the system, with energy reduction of between 30%-50% depending on the type of aggregation.

Measurement work on another real-life testbed implementation found that useful discharge characteristics of the sensor node battery could be obtained over time, and predictions of when batteries reach critical state could be made. Some evidence of battery recovery when devices were left idle was found. The discharge characteristics could also be measured from the perspective of packet transmission numbers, allowing for the possibility of optimising by rationing packet transmissions over a particular period.

For the future, along with Task 2.2, this work feeds into Task 2.3, which aims to assess and group the energy gains between terminal and system level savings. These techniques need to be taken from design techniques and applied in a run-time environment. The energy gains and protocols will be fed into WP3. The mechanisms, and interface requirements need to be forwarded into WP4, and the approaches will form part of the Proof of Concept in WP5.

7. References

[1] INFSO-ICT-257542 CONSERN Project, http://www.ict-consern.eu

[2] Ye, W., Heidemann, J., And Estrin, D. “An energy-efficient MAC protocol for wireless sensor networks”, In Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom), 2000.

[3] Zheng, R.,Hou, J. C., And Sha, L., “Asynchronous wakeup for ad hoc networks”, In Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing. ACM Press, New York, 35–45, 2003.

[4] Polastre, J., Hill, J., And Culler, D., “Versatile low power media access for wireless sensor networks”, In Proceedings of the 2nd International Conference on Embedded Networked Sensor System (SenSys), 2004.

[5] Hohlt, B., Doherty, L., And Brewer, E., “Flexible power scheduling for sensor networks”, In Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN), 2004.

[6] Van Dam, T. And Langendoen, K., “An adaptive energy-efficient mac protocol for wireless sensor networks”, In Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys), 2003.

[7] Zheng, R. And Kravets, R., “On-Demand power management for ad hoc networks”, In Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom), 2003.

[8] Chipara, O., Chenyang Lu, Roman, G.-C., "Efficient Power Management Based on Application Timing Semantics for Wireless Sensor Networks," Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on , vol., no., pp.361-370, 10-10 June 2005.

[9] Chen, B., Jamieson, K., Balakrishnan, H., And Morris, R., “Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks”, In Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom), 2001.

[10] Xu, Y., Heidemann, J., And Estrin, D., “Adaptive energy-conserving routing for multihop ad hoc networks”, Res. Rep. 527, USC. October, 2000.

[11] Xing, G., Lu, C., Zhang, Y., Huang, Q., And Pless, R., “Minimum power configuration in wireless sensor networks”, In Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2005.

[12] Chih-Fan Hsin And Mingyan Liu., “Network coverage using low duty-cycled sensors: random and coordinated sleep algorithms”, In Proceedings of the 3rd international symposium on Information processing in sensor networks (IPSN '04). ACM, New York, NY, USA, 433-442. 2004.

[13] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, Mani Srivastava, "Optimizing Sensor Networks in the Energy-Latency-Density Design Space," IEEE Transactions on Mobile Computing, vol. 1, no. 1, pp. 70-80, Jan.-Mar. 2002

[14] Wikipedia, Profiling (Computer Programming), http://en.wikipedia.org/wiki/Profiling_%28computer_programming%29

[15] Java Programming Language website, http://www.java.com/

[16] Microsoft .NET Framework website, http://www.microsoft.com/net/

[17]Python Programming Language website, http://www.python.org/

[18]Ruby Programming Language website, http://www.ruby-lang.org/en/

[19]N. Nethercote, “Dynamic binary analysis and instrumentation”, Technical Report, ISSN 1476-2986

[20]Wikipedia, Instrumentation, (Computer Programming), http://en.wikipedia.org/wiki/Instrumentation_%28computer_programming%29

[21]Intel® Energy Checker Software Developer Kit User Guide, http://software.intel.com/en-us/articles/intel-energy-checker-sdk/

[22]CONSERN Deliverable 1.1, “Report on Scenarios, Use Cases and System Requirements”

[23]CONSERN Deliverable2.2 “Terminal Level Energy Optimisations Solutions”

[24]CONSERN Deliverable 3.1 “Enablers for Energy-Aware Cooperative Decision and Control”

[25]CONSERN Deliverable 3.2, “Design of Energy-Aware Networking and Cooperation Mechanisms”

[26]CONSERN Milestone 3.2, “Synchronisation with WP1 on Low Energy Protocols”

[27]CONSERN Deliverable 4.1, “Initial Description of Self-Growing Scenarios, Properties, Requirements and Envisaged Framework”

[28]CONSERN Deliverable 4.2, “Distributed Self-Growing Architecture and Interface Description”

[29]CONSERN Milestone 5.1, “Cross-checking of Intermediate Proof-of-Concept Planning with all other WPs”

[30]TinyOS operating system. http://www.tinyos.net/http://www.tinyos.net/

[31]The IBBT W-iLab.t wireless sensor testbed. http://ilabt.ibbt.be/

[32]J. Polastre, J. Hill, D. Culler, “Versatile low power media access for wireless sensor networks”, SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, ACM, New York, NY, USA, 2004, pp. 95-107. doi:http://doi.acm.org/10.1145/1031495.1031508

[33] O. Kummer, F. Wienberg, and M. Duvigneau, "Renew 2.1—User Guide," University of Hamburg, Department for Informatics, Theoretical Foundations Group, Distributed Systems Group, 2006.

[34] W. J. Knottenbelt, "Generalised Markovian analysis of timed transition systems," MSc thesis, Data Networks Laboratory, University of Cape Town, 1996.

[35] J. T. Bradley, N. J. Dingle, S. T. Gilmore, and W. J. Knottenbelt, "Derivation of passage-time densities in PEPA models using ipc: the imperial PEPA compiler," 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS), pp. 344–351, 2003.

[36]N. Akharware, "PIPE2: Platform Independent Petri Net Editor," MSc Degree in Computing Science thesis, Department of Computing, Imperial College of Science, Technology and Medicine (University of London), 2005.

[37]B. Berthomieu, P.-O. Ribet, and F. Vernadat, "The tool TINA—Construction of Abstract State Spaces for Petri Nets and Time Petri Nets," International Journal of Production Research, vol. 42, iss. 14, pp. 2741–2756, 2004.

[38]B. Berthomieu and F. Vernadat, "Time Petri Nets Analysis with TINA," Proceedings of 3rd Int. Conf. on The Quantitative Evaluation of Systems (QEST), Riverside, California, USA, pp. 123–124, 2006.

[39]T. A. Lewis and R. J. Haines, “Formal Verification to Enhance Evolution of Protocols”, Genetic and Evolutionary Computation Conference (GECCO), Montréal, Canada, July 2009.

[40]IEEE 802.15 Working Group for WPAN web page, http://www.ieee802.org/15/, 2010.

[41]IEEE 802.15 WPAN™ Task Group 4 (TG4), http://www.ieee802.org/15/pub/TG4.html, 2010.

[42]ZigBee Alliance web page, http://www.zigbee.org/, 2010.

[43]IPv6 over Low power WPAN (6lowpan), http://datatracker.ietf.org/wg/6lowpan/charter/, 2010.

[44]Routing Over Low Power and Lossy Networks (Roll) web page, http://www.ietf.org/html.charters/roll-charter.html, 2010.

[45]HART Communication Protocol and Foundation - Home Page, http://www.hartcomm.org/, 2010.

[46]Kothmayr, Thomas and Schmitt, Corinna and Braun, Lothar and Carle, Georg, “Gathering Sensor Data in Home Networks with IPFIX”, Technische Universität München Institut für Informatik Garching bei München Germany, Wireless Sensor Networks, Lecture Notes in Computer Science, p.131-146, 2010

[47]Tmote Sky data sheet, http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf