4-2016

Energy efficiency in data collection wireless sensor networks

Miquel Andres Navarro Patino
Purdue University

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Miguel Andres Navarro Patino

Entitled

ENERGY EFFICIENCY IN DATA COLLECTION WIRELESS SENSOR NETWORKS

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Yao Liang
Chair

Zhiyuan Li
Co-chair

Murat Dundar

Christopher Clifton

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Yao Liang

Approved by: William J. Gorman 4/15/2016

Head of the Departmental Graduate Program Date

ENERGY EFFICIENCY IN DATA COLLECTION WIRELESS SENSOR NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Miguel Andrés Navarro Patiño

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2016

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under grants CNS-0758372, CNS-1252066, and CNS-1320132.

First, I would like to thank my advisor Professor Yao Liang for his mentorship and support during my doctoral studies. His guidance and encouragement made this dissertation possible.

I would also like to thank Professor Zhiyuan Li for serving as co-chair in my advisory committee, as well as all other committee members during my preliminary examination and final defense (in alphabetical order) Professor Christopher Clifton, Professor Cristina Nita-Rotaru, Professor Kihong Park, and Professor Murat Dundar. Their insights and research discussions helped me improve this dissertation.

I would like to thank our collaborators from the University of Pittsburgh, Professor Xu Liang, German Villalba, Tyler Davis, and Daniel Salas. Their feedback and efforts with the field deployment and maintenance of sensor nodes significantly contributed to this work.

I would also like to thank my colleagues and fellow PhD students Xiaoyang Zhong, Yimei Li, Halid Yerebakan, and Harold Owens II for their help, discussions, and encouragements in different stages of our PhD studies.

Finally, I would like to thank my parents Amparo and Abelardo, my girlfriend Diana, my sister Olga, and my brother Edgar for their unconditional support, care, and love during all these years.

TABLE OF CONTENTS

Page

LIST OF TABLES v

LIST OF FIGURES vi

ABSTRACT ix

CHAPTER 1. INTRODUCTION 1
1.1 Motivation 1
1.2 Wireless Sensor Networks for Data Collection 2
1.3 Major Contributions 4
1.4 Organization 5

CHAPTER 2. BACKGROUND 6
2.1 TinyOS 6
2.2 The Collection Tree Protocol 7
2.2.1 Main Components 11
2.3 Low Power Listening (LPL) 15

CHAPTER 3. THE ASWP TESTBED 18
3.1 Deployment 18
3.2 Preliminary Performance Analysis 23

CHAPTER 4. ENERGY EFFICIENT AND BALANCED ROUTING (EER) 28
4.1 Related Works 29
4.2 Design of EER 33
4.2.1 Energy Efficiency 33
4.2.2 Method 34
4.2.3 Implementation 36
4.2.4 Parent Set Size for Network Diagnosis 41
4.3 Analytical Performance Model 42

CHAPTER 5. EVALUATION OF EER 50
5.1 Experiments and Simulations 50
5.1.1 Experiments in Indriya 52
5.1.2 Simulations in Cooja 57
5.1.3 Discussion 60
5.2 Case Study: ASWP WSN Testbed 62
5.2.1 WSN Application Description 62
5.2.2 Protocol Evaluation 62

Page

CHAPTER 6. ENERGY PROFILES 68
6.1 Related Work 68
6.2 Method 70
6.3 Experiments 72
6.4 Results 73
6.4.1 Energy Profiles 76
6.4.2 Node Lifetime 80

CHAPTER 7. INTEGRATED NETWORK AND DATA MANAGEMENT SYSTEM FOR HETEROGENEOUS WSNS 83
7.1 Related Works 85
7.2 Management System Architecture 86
7.2.1 Layered Architecture 87
7.2.2 Components View 88
7.2.3 Agent Functions 90
7.2.4 User Access Control 90
7.3 Agent-Server Communication 92
7.3.1 Agent-Server Protocol 93
7.3.2 Unified Gateway (UG) Web Service 95
7.4 Data Monitoring 97
7.5 System Implementation 99
7.5.1 Agent for XServe 99
7.5.2 Agent for TinyOS 100
7.6 Deployment and Web Interface 101
7.7 Data Functions 104
7.7.1 Data Indicators 105

CHAPTER 8. SUMMARY AND FUTURE WORK 109
8.1 Summary 109
8.2 Future Work 112

REFERENCES 114

APPENDIX. PACKET FORMAT AND APPLICATION FOR CTP+EER 122

VITA 124

PUBLICATIONS 125

LIST OF TABLES

Table		Page
3.1	Characteristics of related WSN deployments	20
3.2	Datasets for preliminary analysis	24
3.3	Network average PRR of XMesh and CTP	26
4.1	Parameters of the analytical model of CTP+EER	43
5.1	Average performance of CTP+EER versus CTP and ORW in Indriya	53
5.2	Summary of results from simulations with 20 WSN nodes	58
5.3	Summary of results for CTP+EER from the ASWP WSN testbed	63
6.1	Electric current measurements	75
6.2	Mote parameters and configurations	76
6.3	Expected node lifetime from laboratory experiments	81
6.4	Expected lifetime from nodes at the ASWP testbed	82

LIST OF FIGURES

Figure		Page
2.1	An example of the iterations to compute the node ETX in CTP.	8
2.2	CTP data packet structure.	9
2.3	CTP routing packet structure.	10
2.4	An example of BoX-MAC in TinyOS for a data packet transmission from node A to node B.	16
3.1	Location of the ASWP testbed (red dot) in Allegheny County (highlighted in cyan and enlarged), Pennsylvania, USA.	18
3.2	Examples of node configurations in the ASWP testbed. A node hanging from a tree without external sensors (left) and a node mounted onto a PVC pipe with external sensors attached (right).	19
3.3	Location of the 84 WSN nodes in the ASWP WSN testbed as of August 2015.	21
3.4	Limited sun exposure in locations of interest at the ASWP testbed.	22
3.5	Summary of results obtained from XMesh at the ASWP testbed during the complete time period. Daily averages are shown for each node, solid lines indicate the network average, and node IDs are sorted based on their distance to the sink.	25
3.6	Summary of results obtained from CTP at the ASWP testbed during the complete time period. Daily averages are shown for each node, solid lines indicate the network average, and node IDs are sorted based on their distance to the sink.	27
4.1	An example of the parent set of a sending node x with primary parent node P.	35
4.2	Main components of CTP+EER.	37
4.3	General flowchart of the packet forwarding process in CTP+EER.	39
4.4	An example of a routing cost inconsistency in EER without relaxing the loop-detection condition.	40

Figure Page

4.5 A general illustration of the subtrees of descendants where the largest blue subtree is the worst case built in CTP (solid lines) for neighbors of the sink. Dashed lines represent additional links used in CTP+EER that distribute the traffic out of the original subtree in CTP. 45

5.1 Path length distribution in Indriya based on CTP. 52

5.2 Results from experiments in Indriya. 54

5.3 Representation of the network topology of experiments using CTP+EER in Indriya. 56

5.4 Improvement of the transmission cost and duty cycle in CTP+EER compared to CTP in simulations with random topologies in Cooja. 60

5.5 Results from the evaluation of CTP+EER in the ASWP testbed. . . . 64

5.6 Location of the 84 WSN nodes in the ASWP WSN testbed as of August 2015. Weak nodes diagnosed by CTP+EER are highlighted in orange. 66

6.1 Data packet generation for an IRIS node with ADCs enabled on the MDA300 acquisition board: (1) no external sensors (top); (2) two external soil moisture sensors (middle); (3) three external soil moisture sensors (bottom). 74

6.2 Traffic characteristics of selected nodes from the ASWP testbed. Generated data packets (Ge), forwarded/received data packets (Fw), data packet retransmissions (rTx), routing packet transmissions (cTx), and routing packet receptions (cRx). Daily average values are presented. 77

6.3 Energy profiles and duty cycles of selected nodes at ASWP. Daily average values are presented for the following application states: data sampling, data transmissions (DataTx), data receiving+forwarding (RxFw), data retransmission (ReTx), routing transmissions (ctlTx), routing receptions (ctlRx), idle listening, and sleeping. 78

6.4 Results obtained for nodes in a laboratory experiment. Daily average values are presented. Drivers are enabled in these nodes, but the external sensors are not attached. They use two AA batteries of the same reference as nodes at ASWP. 80

7.1 An illustration of the management system general architecture. 84

7.2 INDAMS layered architecture. 87

7.3 INDAMS components view. 89

7.4 Control/Data Handler and server Unified Gateway (UG). 90

Figure Page

7.5 Graphical representation of function categories at the UGL for a generic client/server scenario. 91

7.6 Data model for user access control in INDAMS. 92

7.7 Simplified state transition diagram of an agent. 95

7.8 Simplified state transition diagram of the server. 96

7.9 Implementation example of the agent-server communication via web services. 97

7.10 Multiple clients trying to monitor the same WSN. The data collection function (orange) goes through the UGL and it is executed only once for each WSN. The data monitoring function does not go through the UGL and each client receives the collected data. 98

7.11 Operations of the data handler. 99

7.12 An illustration of INDAMS for WSN topology monitoring in a residential backyard. 101

7.13 INDAMS web interface for agent selection. 102

7.14 An example of the agent functions available for the ASWP testbed. . . 103

7.15 Topology monitoring for the ASWP testbed. 103

7.16 Data monitoring for the ASWP testbed. 105

7.17 Topology monitoring and status of nodes at the ASWP testbed with 84 nodes. Node colors indicate their batteries levels: charged (green), depleted (gray), and close to be depleted (orange). 106

7.18 Web interface of the database query and export function in INDAMS. . 107

7.19 Web interface of the last received packet function in INDAMS. 108

7.20 Web interface for the data indicator function in INDAMS. 108

A.1 CTP+EER basic data packet structure for network diagnosis. 122

ABSTRACT

Navarro Patiño, Miguel A. PhD, Purdue University, May 2016. Energy Efficiency in Data Collection Wireless Sensor Networks. Major Professor: Yao Liang.

This dissertation studies the problem of energy efficiency in resource constrained and heterogeneous wireless sensor networks (WSNs) for data collection applications in real-world scenarios. The problem is addressed from three different perspectives: network routing, node energy profiles, and network management. First, the energy efficiency in a WSN is formulated as a load balancing problem, where the routing layer can diagnose and exploit the WSN topology redundancy to reduce the data traffic processed in critical nodes, independent of their hardware platform, improving their energy consumption and extending the network lifetime. We propose a new routing strategy that extends traditional cost-based routing protocols and improves their energy efficiency, while maintaining high reliability. The evaluation of our approach shows a reduction in the energy consumption of the routing layer in the busiest nodes ranging from 11% to 59%, while maintaining over 99% reliability in WSN data collection applications. Second, a study of the effect of the MAC layer on the network energy efficiency is performed based on the nodes energy consumption profile. The resulting energy profiles reveal significant differences in the energy consumption of WSN nodes depending on their external sensors, as well as their sensitivity to changes in network traffic dynamics. Finally, the design of a general integrated framework and data management system for heterogeneous WSNs is presented. This framework not only allows external users to collect data, while monitoring the network performance and energy consumption, but also enables our proposed network redundancy diagnosis and energy profile calculations.

CHAPTER 1. INTRODUCTION

1.1 Motivation

Advances in semiconductor technologies allow electronic devices, with a given computing capacity, to decrease their size and cost at an exponential rate, following Moore's law. Researchers have been able to use these semiconductor-manufacturing techniques for building smaller and inexpensive radios, sensors, and actuators, which enable new possibilities for instrumenting and interacting with the physical world [1]. Wireless sensor networks (WSNs) integrate these low-cost, low-power, and multi-functional devices, presenting a promising approach for various sensing and actuating tasks in multiple application domains.

Unlike traditional networks, WSNs are formed from resource-constrained devices, with limited processing capacity, memory, communication bandwidth, and energy sources [2]. These distinctive characteristics allow new applications but also introduce new challenges, unique to sensor networks, compared to other kinds of wireless networks. First, resource limitations are substantial at the node level. For this reason, protocols and algorithms running in WSNs must involve node cooperation to overcome such limitations. Moreover, programs executing in WSN nodes must comply with memory constraints, and therefore, WSN applications require complex design approaches, i.e., operating systems customized for the needs of each particular application [3] [4].

WSNs are envisioned to operate at high scales in terms of size (i.e., number of nodes) and deployment time. Low hardware costs facilitate the acquisition of WSN nodes, not to mention all of them need not use the same components or sensors. In consequence, protocols and WSN applications must be designed considering a high number of nodes and hardware heterogeneity.

WSNs have been used in a variety of applications, e.g., military surveillance, forest fire detection, industrial automation, environmental monitoring, health applications, among others [5] [2]. In general, these applications can be classified into two major categories: *data collection* and *object detection/tracking*. Data collection applications involve periodic sensing and are designed to sample and transmit their sensor data at defined time intervals. This category normally includes applications with low duty cycles and high network lifetime. The second category corresponds to object detection/tracking applications. These applications are designed for reacting after an external event, i.e., forest fire or intrusions, and thus their first objective is detection followed by tracking the event behavior. These processes usually involve active coordination between WSN nodes, and therefore, this category follows a very different approach compared to data collection applications. For object detection and tracking, faster sampling rates are expected and accuracy is often preferred at the expense of higher energy consumption. This classification is not absolute and there are multiple scenarios in which WSN applications require incorporating behaviors from both categories.

1.2 Wireless Sensor Networks for Data Collection

This dissertation is focused on WSNs for data collection, which have emerged as a promising alternative to traditional data-collection mechanisms (i.e., data loggers and sensing stations) enabling cost-effective implementations in various sciences and engineering domains. In this context, WSN nodes are typically deployed outdoors in harsh environments, which pose great challenges for multi-hop WSN deployments.

WSN nodes are battery powered and thus energy availability is a significant restrictive factor, in addition to hardware heterogeneity and memory, computing, and bandwidth limitations. In some cases, energy constraints in WSN nodes can be mitigated by the use of energy harvesting mechanisms [6] [7] [8], whereas in many other situations WSN deployments have to rely on batteries as their main energy

source [9] [10] [11] [12] [13] (e.g., due to space constraints, limited sun exposure), presenting an urgent need for energy efficiency.

Previous studies have attempted to address these energy efficiency issues based on cross-layer designs [14] [15], limiting their practical applications because of the complexity of re-implementing and replicating the original cross-layer dynamics in other hardware platforms. Likewise, studies oriented towards energy efficient MAC layer implementations face similar challenges in the presence of heterogeneous WSN nodes.

The evaluation of WSN protocols and applications also provide some challenges, as different implementations cannot be tested under the exactly same real-world and complex environment. Up to date, different methods have been used to approach these challenges, including theoretical model formulations, simulations, emulations, and experimental testing. Theoretical models enable the mathematical derivation of general or asymptotic conditions; although in these cases environment dynamics are often simplified to avoid excessive model complexity. A similar drawback is presented in simulations, where even though the same environment can be used to evaluate multiple approaches, it corresponds to a modeled environment that cannot capture the complex dynamics of real scenarios. Emulations go one step further by allowing to evaluate both algorithms and implementations simultaneously, but still under simplified environment conditions. Additional information can be obtained from experiments in WSN motes, e.g., using publicly available WSN testbeds; however, real-world deployments are also required for validating data collection applications targeting outdoor environments. Such real-world experiments consider all factors involved in these deployments, which can be easily omitted in any of the previous scenarios. In this dissertation, theoretical models, simulation/emulation, as well as indoor and outdoor testbed experiments are used throughout the validations.

1.3 Major Contributions

This dissertation studies the problem of energy efficiency in resource constrained and heterogeneous WSNs for data collection applications in real-world scenarios. This problem is addressed from three different perspectives: network routing, node energy profiles, and network management.

First, the energy efficiency problem is formulated as a load balancing problem where the routing layer can exploit the network redundancy offered by the WSN topology to reduce the data traffic processed in critical nodes, improving the energy consumption and network lifetime. This approach is independent of the hardware platform and reduces the data traffic load on critical nodes by carefully introducing suboptimal paths, which results in an overall cost-effective solution that extends traditional cost-based routing protocols, without routing overhead. In addition, the resulting routing strategy is able diagnose nodes with low network redundancy, allowing network administrators to correct these situations improving the network energy efficiency and also preventing network partitions in the event of battery depletion or node failures. The evaluation of our approach shows a reduction in the energy consumption of the routing layer in the busiest nodes ranging from 11% to 59%, while maintaining over 99% reliability in data collection applications

Then, a study of the effect of the MAC layer on the network energy efficiency is performed by analyzing the energy consumption profiles of WSN nodes in real-world scenarios. This approach combines health and instrumentation information from deployed WSN nodes with electric current measurements for each state of the WSN application, reflecting the effect of the external environment and network dynamics. Such measurements can be obtained in a laboratory beforehand for a specific MAC layer implementation, and thus energy profiles can be directly analyzed on WSN deployments. The resulting energy profiles reveal significant differences in the energy consumption of WSN nodes depending on their external sensors, as well as

their sensitivity to changes in network traffic dynamics, resulting in higher energy consumption.

Finally, the design and implementation of an integrated framework for network and data management in WSNs is presented. This framework systematically supports heterogeneous WSNs under a unified management system and separates management from application functionalities. Furthermore, by processing the health and instrumentation data collected from WSNs, the framework enables the above mentioned network redundancy diagnosis, as well as the energy profile calculations. This information is available to network administrators who can take appropriate actions to prevent undesired network behaviors, or react to specific events.

1.4 Organization

This dissertation is organized as follows. Chapter 2 describes the background on WSNs for data collection. Chapter 3 introduces our outdoor WSN testbed deployment. Chapter 4 presents our new energy efficient and balanced routing strategy and its implementation. Chapter 5 shows the evaluation of this routing strategy and a case study of its deployment in a real-world outdoor WSN testbed. Chapter 6 presents the construction and analysis of the energy profiles. Chapter 7 gives the design and implementation of the framework for network and data management. Finally, Chapter 8 presents the summary of the current work and outlines the future work.

CHAPTER 2. BACKGROUND

As our work is focused on practical WSNs, this chapter introduces the necessary tools for developing data collection WSN applications. It presents the operating system that is used in this dissertation, followed by the standard implementation of routing and MAC layers.

2.1 TinyOS

TinyOS [4] is the most widely used operating system for WSNs. It defines a component-based framework, where WSN applications select a subset of components to build an application-specific operating system into each application. Applications in TinyOS use the NesC language [16], and their typical size is of a few kilo bytes, of which the operating system base is around 400 bytes in RAM.

The components that define a TinyOS program are connected through *interfaces*, and use the following computational abstractions: *commands*, *events*, and *tasks*. Requests of services between components are performed through *commands* (e.g., transmit a data packet), and the corresponding completion of the service is signaled with an *event* (e.g., transmission done).

For increasing the system responsiveness, TinyOS allows components to defer time consuming computations using *tasks*. Tasks constitute functions that are executed by the TinyOS scheduler at a later time following a run-to-completion execution model. Therefore, tasks can perform background computation and cannot be preempted by other tasks, although they can still be preempted by asynchronous code (i.e., interrupt handlers, commands and events).

The amount of work carried by components and events is reduced through *split-phase operations*, where the service request (i.e., *commands*) and the completion

signal (i.e., *events*) are decoupled. In this way, a command can post a task with long-running operations and return immediately. Later, the TinyOS scheduler executes the task and once the task finishes an event is signaled.

Throughout this dissertation TinyOS v2.1.2 is used. Other operating systems for WSNs include Contiki [17], RIOT [18], and OpenWSN [19].

2.2 The Collection Tree Protocol

The collection tree protocol (CTP) [20] [21] is the de-facto standard routing protocol for WSNs. It is designed to maintain a robust operation in data collection WSN applications, while promptly reacting to topology changes. The protocol combines three techniques: (1) it uses a link estimator for computing the link quality to neighbor nodes, using information from both data and routing packets; (2) CTP uses the Trickle algorithm [22] for timing routing packets and adapting to different network conditions; and (3) CTP performs datapath validation for detecting and recovering routing loops.

CTP is a cost-based routing protocol that aims to compute the best available path from a WSN node to a sink. In the protocol, cost information is disseminated using routing packets and path computations are performed in an iterative manner, as in distributed distance-vector routing protocols, using the expected number of transmissions (ETX) [23] as cost metric. ETX values are associated to links (i.e., *link ETX*) and nodes (i.e., *node ETX*), where the ETX of a node corresponds to the sum of link ETX values in the best path towards the sink, and the sink node has a default ETX equal to zero. An example of the iterations to compute node ETX values in CTP is shown in Figure 2.1, where nodes located one hop from the sink compute their costs based on their link qualities, knowing that the sink has a node ETX equal to zero. In the next iteration, nodes two hops from the sink compute their node costs and the process continues until all nodes in the network define their node ETX values.

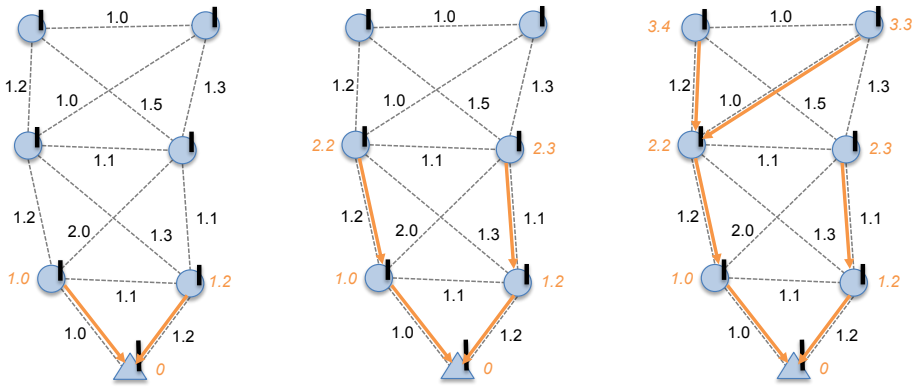


Figure 2.1. An example of the iterations to compute the node ETX in CTP.

The architecture of CTP defines three major components: *Link Estimator*, *Routing Engine*, and *Forwarding Engine*. These components run on each sensor node and they are connected through multiple interfaces. The Link Estimator computes and maintains the link cost of neighbor nodes. The link ETX is computed taking into account both inbound and outbound link qualities, which are then passed through an exponential smoothing filter. Inbound link quality is computed based on routing packets and outbound link quality is based on data packet transmissions and their acknowledgements. The Routing Engine controls routing packet transmissions based on the Trickle algorithm [22]. It manages the routing table with node ETX values, and it is also in charge of selecting the parent node. The Forwarding Engine is in charge of forwarding data packets, either generated by the sending node or received from its neighbors. It controls data packet retransmissions and indicates the Link Estimator when to update the outbound link quality in the event of packet loss. It also performs loop detection, identifying packets received from nodes with lower ETX as inconsistencies. When this occurs, new routing packets are requested from neighbor nodes, through the Routing Engine, to update the local information before attempting to forward data packets.

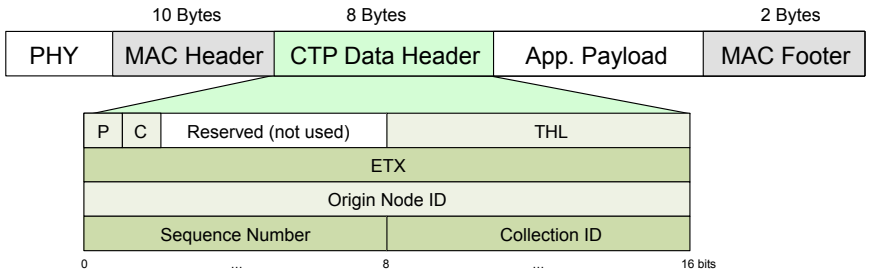


Figure 2.2. CTP data packet structure.

Packet Structures

Data Packets: the structure of a generic data packet is presented in Figure 2.2. Physical and MAC-layer headers add a fixed overhead to every data packet transmission. In addition, the MAC layer also appends a 2-Byte footer used for error detection. Following the MAC-layer header, there is the *CTP data header* (8 Bytes), and then, the application data payload is introduced.

The *CTP data header* defines the *Pull (P)* and *Congestion (C)* flags. The remaining 6 bits of the first byte are not specified and they are reserved for future usage. The second byte includes the *Time Has Lived (THL)* metric, which works as a hop counter. Third and fourth bytes defines the *ETX* of the sending node, which is used for loop detection. This value is updated after each hop until the sink node is reached. The node ID of the original sending node is included in the *origin node ID* field on bytes fifth and sixth. The byte in position seven contains the *CTP sequence number* for data packets. Finally, the eighth byte defines the *collection ID* value, which indicates the instance of CTP intended to handle the current data packet, as multiple collection services may be initialized from the application layer [24].

Routing Packets: these packets include specific headers controlled by the Link Estimator and Routing Engine components, as shown in Figure 2.3. The routing packet structure shows that a fixed-size header from the Link Estimator (2 Bytes) and a CTP routing header (5 Bytes) are included. After these, there is a variable-size

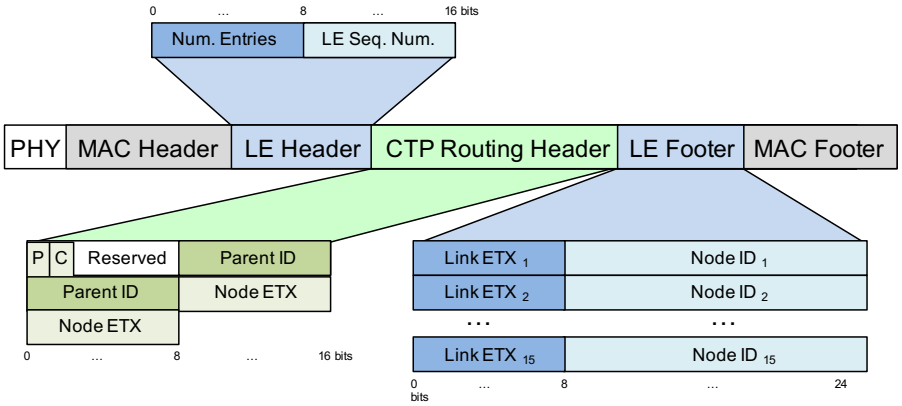


Figure 2.3. CTP routing packet structure.

footer from the Link Estimator, which may be from 0 to 45 Bytes. Routing packets are processed by the MAC and physical layers in the same way as data packets, and therefore, they also incur in the same header and footer overheads.

The header from the Link Estimator defines two bytes. The first byte is used for coding the number of elements that will be added into the Link Estimator footer, and the second byte defines the *LE sequence number*, which is used for computing the inbound link quality.

The CTP routing header requires five bytes in routing packets. The first byte defines the same entries as the CTP data header described earlier. In addition, the remaining 4 bytes are used for specifying the *parent ID* of the current node (2 Bytes) and the *node ETX* (2 Bytes) (i.e., path ETX through the parent node).

Finally, the footer from the Link Estimator defines *neighbor node ID - link ETX* pairs, which are appended to routing packets depending on the number of neighbors. *Link ETX* values in this footer indicate the link cost from the sending node to the neighbor node.

2.2.1 Main Components

This section provides a more detailed description of the implementation of CTP components in TinyOS, based on [21] [24] [25].

Link Estimator

The implementation of CTP available in TinyOS uses the 4-bit link quality estimator [26]. The Link Estimator (LE) component introduces a header and a footer in routing packets, and it also provides the required interfaces to the Routing Engine and Forwarding Engine components. All values computed in the LE component are stored in a *neighbor table*, different from the *routing table* maintained by the Routing Engine.

For computing the link ETX value to each neighbor, both transmission (i.e., outbound) and reception (i.e., inbound) link qualities are considered. Transmission quality is based on data packet transmissions ($dataPkt_{tx}$) and the acknowledgments received for these packets ($dataPkt_{ack}$), as defined in (2.1). These values are used after a specific number of data packets are transmitted to complete a predefined packet window.

$$Q_{tx} = \frac{dataPkt_{tx}}{dataPkt_{ack}} \tag{2.1}$$

Reception quality is computed based on the number of routing packets received from each neighbor. As shown in (2.2), reception quality is defined as the ratio between the number of routing packets received ($routingPkts_{rx}$) and the total number of routing packets sent from a neighbor node ($routingPkts_{tx}$). Each node advertises the number of routing packets transmitted using the *LE sequence number* field in routing packets (see Figure 2.3). Similar to transmission quality, reception quality is also used after a specific number of beacons are received from a neighbor node. Before calculating the link ETX, the reception quality estimate is passed through an exponential smoothing filter, averaging new values with previous samples weighted

with an exponentially decaying function [24]. The reception quality estimate used for computing the link ETX is shown in (2.3) and the default value of the decay factor is 0.9.

$$Q_{rx} = \frac{routingPkt_{rx}}{routingPkt_{tx}} \tag{2.2}$$

$$Q_{rx}[t] = \alpha\frac{routingPkt_{rx}}{routingPkt_{tx}} + (1-\alpha)Q_{rx}[t-1] \tag{2.3}$$

Finally, the link ETX for a specific neighbor is computed as defined in (2.4), where Q represents either transmission or reception quality estimates. As a result, the link ETX value is updated every time a new transmission or reception quality estimate is available, using another exponential smoothing filter with default decay factor of 0.9.

$$LinkETX = \alpha Q + (1-\alpha)LinkETX_{old} \tag{2.4}$$

Routing Engine

As mentioned above, the Routing Engine (REng) in CTP is in charge of handling routing packets, selecting the parent node and maintaining the routing table. The time interval for routing packets is managed based on the Trickle algorithm [22], which basically doubles the time interval after each routing packet transmission until a maximum threshold is reached. CTP defines three possible causes for resetting the Trickle timer:

- *Pull bit:* when nodes have no routes (e.g., after booting), they set the Pull bit in the CTP header (see Figure 2.2 and Figure 2.3). In response, when nodes receive a packet with the Pull bit set, they reset their Trickle timer and thus routing traffic increases allowing nodes to establish new routes.

- *Routing Loops:* when a routing loop is detected by the Forwarding Engine component, it also triggers a Trickle timer reset. By resetting his Trickle timer,

the node detecting a loop sends more frequent routing packets and other nodes in the loop will be able to update their routing information. More about routing loops will be discussed with the Forwarding Engine component.

- *ETX changes:* when a node detects a sudden improvement in its node ETX value greater than a threshold, it resets its Trickle timer, triggering new route updates and informing neighbor nodes about this improvement. The default threshold is 2.0.

One additional condition for resetting the Trickle algorithm was defined in [27]. The condition defines that any node in the network can reset their Trickle timer if they receive a routing packet from a neighbor node with a very high node ETX (e.g., higher than 10). In the event of temporary network partitions, data packet retransmissions and routing loops will cause the node ETX from disconnected nodes to increase continuously in a very short time period. Since connected nodes are not experiencing any of the previous three conditions, their Trickle timer will continue to increase exponentially, delaying the opportunity for neighbor nodes to reconnect to the network. By resetting the Trickle timer when high node ETX values are detected from neighbors, nodes that were temporarily disconnected will be able to rejoin the collection tree faster. Similar to the previous condition, this also corresponds to an efficiency optimization, which allows CTP to react faster to topology changes, unlike the first two conditions, which are necessary for correctness [21].

The REng in CTP implements the following conditions for selecting a new parent node:

- If the current parent node is congested and the path ETX through a neighbor node is strictly lower than the sending node ETX (i.e., path ETX through the current parent), plus a threshold of 1.0, then the neighbor node is selected as a new parent node. This condition is presented in (2.5) and it indicates that the new parent node is selected avoiding children from the current node. However,

this condition is never used because congestion has not been implemented in CTP for TinyOS.

- If the path ETX through a neighbor node plus a parent-switch threshold is strictly less than the node ETX (i.e., path ETX through the current parent), then the neighbor node is selected as the new parent. By default, the parent-switch threshold in CTP for TinyOS is 1.5, and this condition is presented in (2.6).

$$parent_{congested} \quad \& \quad (PathETX_{bestNeighbor} < NodeETX + 1.0) \tag{2.5}$$

$$PathETX_{bestNeighbor} + 1.5 < NodeETX + LinkETX_{toParent} \tag{2.6}$$

Forwarding Engine

All data packets generated by the application layer or received from neighbor nodes are handled by the Forwarding Engine (FEng). First, packets are inserted into a forwarding queue, where they are processed in a first-come, first-served basis. To this end, the FEng retrieves the ID of the current parent from the REng, then, the FEng requests an acknowledgement and passes the data packet to the MAC layer. If the acknowledgement is received after the packet transmission, the data packet is removed from the queue; otherwise, the FEng triggers a CTP retransmission.

CTP retransmissions are performed independently of any MAC-layer retransmissions. After a packet acknowledgement is lost, the FEng asks the REng to re-compute routes. This process triggers a new parent selection procedure, but it does not necessarily reset the Trickle timer, unless one of the previous conditions is met. Then, the FEng activates a back-off timer and retries sending the data packet to the MAC-layer. This process is repeated until a maximum number of retransmissions is reached, in which case the data packet would be discarded. The default maximum number of retransmissions in CTP is 30 attempts.

Routing loops in CTP are detected based on inconsistencies of the node ETX values. For the FEng, an inconsistency occurs in data packets received from neighbor nodes, if the received node ETX is lower or equal than the node ETX of the current node. In this case, the FEng triggers a Trickle reset in the REng, generating new routing packets. Moreover, the FEng does not drop looping packets. Instead, it introduces a delay before attempting to forward the packet, allowing other nodes to fix their routes before the transmission. Allowing packets to traverse multiple loops also intends to incrementally repair the network topology and if loops are rare events, then the cost of looping packets would remain low [21].

An additional feature of the FEng is duplicate packet control. Duplicate packets are identified based on their *Origin ID*, *Collection ID*, *CTP Sequence Number* and *THL*. By adding the THL value, CTP is only considering 1-hop duplicates and avoids discarding looping packets. The FEng maintains a cache of default size 4, which stores the last successfully transmitted packets.

2.3 Low Power Listening (LPL)

TinyOS provides a MAC layer abstraction that implements an asynchronous low power listening (LPL) strategy to duty cycle the radio, reducing the energy consumption in WSN nodes [28]. In LPL, nodes sleep most of the time and periodically wake up to sample the wireless channel for activity. When energy is detected, nodes stay awake to receive incoming packets, or they go back to sleep after a predefined timeout interval. In this strategy, most of the work is performed by sending nodes, since nodes in the network know each others *wakeup interval*, but they do not know their wakeup times. Therefore, to guarantee that the intended receiving node is able to hear the packet transmission, the sending node needs to transmit during the entire *wakeup interval* [29].

TinyOS implements BoX-MAC [30] for LPL, which uses a continuous transmission of the data packet for the duration of the wakeup interval, replacing preamble trans-

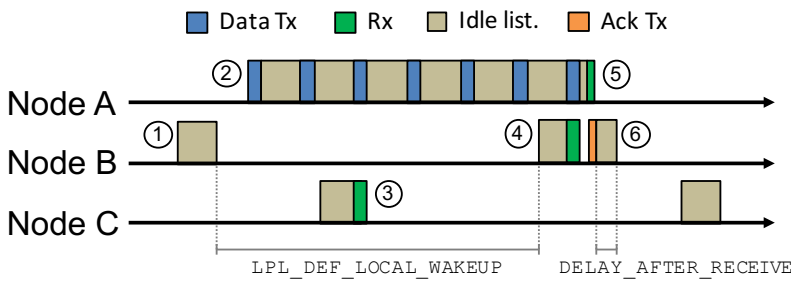


Figure 2.4. An example of BoX-MAC in TinyOS for a data packet transmission from node A to node B.

missions used in previous LPL definitions. In this way, both sending and receiving nodes can save additional energy, as receiving nodes are now able to quickly identify packets destined to them and go back to sleep if necessary, while sending nodes can finish a data packet transmission early if an acknowledgement is received before the wakeup interval is completed. Figure 2.4 shows an example of BoX-MAC in TinyOS for a data packet transmission from node A to node B. The figure indicates different events as follows: (1) node B wakes up periodically to sample the wireless channel; (2) node A starts a data packet transmission with node B as destination; (3) node C wakes up, detects activity in the channel, and receives the data packet from A, which is quickly discarded after verifying that a different node is the destination; (4) node B wakes up and receives the data packet from A; (5) node B sends the corresponding packet acknowledgement, node A receives it and finishes the data packet transmission; (6) after sending the packet acknowledgement, node B continues listening the channel for a short time in case more packets were being transmitted.

Parameters that need to be controlled by the LPL implementations in TinyOS are the local wakeup interval (`LPL_DEF_LOCAL_WAKEUP`), the neighbors' wakeup interval (`LPL_DEF_REMOTE_WAKEUP`), and the time a receiving node will continue listening after receiving the data packet (`DELAY_AFTER_RECEIVE`), as seen in Figure 2.4. All sleeping nodes in the network configure their local wakeup interval equal to their neighbors' wakeup interval, while nodes that must remain awake (e.g., sink) set their

local wakeup interval to zero. Other important parameters include the time a sending node waits for a packet acknowledgement (i.e., time interval between consecutive packet transmissions as seen in the second event of Figure 2.4), the time that nodes spend sampling the wireless channel, and the time a node requires to receive and process a packet acknowledgement. The proper configuration of these timing parameters is critical, since if nodes do not spend enough time sampling the channel, they may wake up and go back to sleep while the sending node is still waiting for packet acknowledgments, resulting in unnecessary packet retransmissions. Likewise, if the sending node does not wait long enough between consecutive packet transmissions, it may miss the packet acknowledgement, resulting in unnecessary retransmissions and duplicate packets. It is also important to note that the time required to receive a packet acknowledgement depends on each specific hardware platform, increasing the complexity of heterogeneous WSNs.

TinyOS has two different architectures for LPL: the *default LPL* implementation and the `rfxlink` implementation. The default LPL implementation is available for the CC2420 driver in `tos/chips/cc2420/`. It controls the time that a node samples the channel with the number of clear channel assessment (CCA) samples defined by the parameter `MAX_LPL_CCA_CHECKS`. On the other hand, the `rfxlink` implementation aims to provide a general architecture of the transceiver driver. It controls the LPL parameters with the `LowPowerListeningConfig` interface. The RF230 is one of the transceivers with a driver directly developed using the `rfxlink` architecture. The CC2420 transceiver also has a driver with this architecture available in `tos/chips/cc2420x/`; however, unstable behaviors were observed in preliminary tests for different configurations of this driver, and thus it is not used in this dissertation.

CHAPTER 3. THE ASWP TESTBED

This chapter introduces the WSN testbed deployment that defines the objective scenario of our work with preliminary performance results.

3.1 Deployment

The testbed is deployed at the Beechwood Farms Nature Reserve (BFNR) of the Audubon Society of Western Pennsylvania (ASWP), located in Fox Chapel in northern Allegheny County, Pennsylvania, USA, as shown in Figure 3.1. The BFNR is 134 acres of protected land, which is owned by the Western Pennsylvania Conservancy.

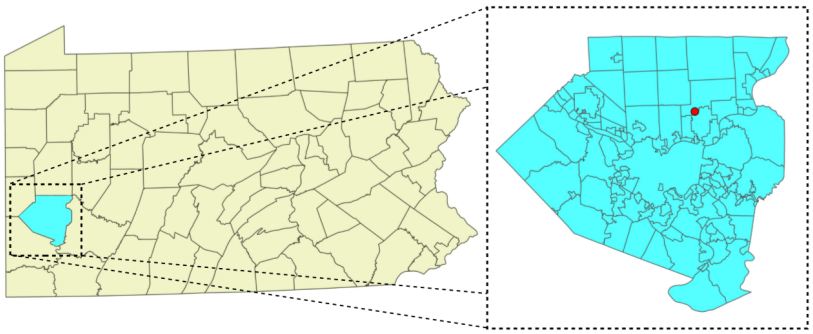


Figure 3.1. Location of the ASWP testbed (red dot) in Allegheny County (highlighted in cyan and enlarged), Pennsylvania, USA.

The ASWP testbed was initially deployed in 2010 [31] and since then it has been operating with the objective of exploring the feasibility and challenges of using WSNs for collecting reliable long-term hydrological data and investigating the impacts of vegetation heterogeneity and soil properties on the status and trends of soil moisture and transpiration. Table 3.1 summarizes representative WSN deployments reported in

the past decade, specifying their deployment analysis time, network size, deployment environment, platform, and main application category.

The hardware in the testbed combines heterogeneous WSN platforms and all nodes perform both networking and sensing tasks in a flat network setting (i.e., non-hierarchical deployment). As of August 2015, the ASWP testbed includes 84 WSN nodes, which correspond to 35 TelosB, 25 MICAz, and 24 IRIS motes, with one IRIS sink node. TelosB motes incorporate the 2.4 GHz CC2420 transceiver and the MSP430 microcontroller with 10 KB of RAM. MICAz motes have the same CC2420 transceiver as TelosB motes, and an ATmega128L microcontroller with 4 KB of RAM. The IRIS platform uses the 2.4 GHz RF230 transceiver, and the ATmega1281 microcontroller with 8 KB of RAM. It should be noted that the RF230 transceiver of IRIS motes has a higher maximum output power of 3 dBm, compared to 0 dBm in the CC2420 transceiver of TelosB and MICAz motes.

Figure 3.2. Examples of node configurations in the ASWP testbed. A node hanging from a tree without external sensors (left) and a node mounted onto a PVC pipe with external sensors attached (right).

The BFNR landscape and facility present unique challenges for the deployment of WSN nodes. Internet access is only available at the BFNR Nature Center, while the locations of interest to install the sensors are around 300 m away or farther. In

Table 3.1.
Characteristics of related WSN deployments

	Deployment Analysis Time	**Size**	**Environment**	**Platform**	**Application**
MoteLab [32]	N/A	190 Nodes	Indoors	TMote Sky	App. testing
Kansei Genie [33]	N/A	700 nodes	Indoors	XSM,TelosB,Imote2	App. testing
Indriya [34]	N/A	139 nodes	Indoors	TelosB	App. testing
SensLab [35]	N/A	256x4 nodes	Indoors	WSN430	App. testing
FlockLab [36]	N/A	30 nodes	Indoors	IRIS,TelosB,Opal,Tiny	App. testing
VigilNet [37]	days	70 nodes	Outdoors/open area	MICA2	Tracking/detection
ExScal [38]	15 days	1200 nodes	Outdoors/open area	MICA2,XSM	Tracking/detection
GreenOrbs [12]	29 days	330 nodes	Outdoors/forest	TelosB	Data collection
SNF [7]	30 days	57 nodes	Outdoors/forest	M2135	Data collection
Redwoods [10]	44 days	33 nodes	Outdoors/tree	MICA2Dot	Data collection
SensorScope [6]	2 months	97 nodes (16 outdoors)	Outdoors/glacier	TinyNode	Data collection
Trio [39]	4 months	557 nodes	Outdoors/open area	Trio Mote	Tracking/detection
GDI [9]	4 months	98+49 nodes	Outdoors	MICA2Dot	Data collection
LUYF [40]	147 days	10 nodes	Outdoors	MICAz	Data collection
ASWP	>1 year	84 nodes	Outdoors/forest	MICAz,IRIS,TelosB	Data collection

addition, the aesthetics of the nature reserve must be maintained, and thus, there are limited locations available where WSN nodes can be placed, especially in areas closer to the Nature Center. For this purpose, the node enclosures were camouflaged and discretely located either hanging from tree branches or mounted onto PVC pipes, as seen in Figure 3.2.

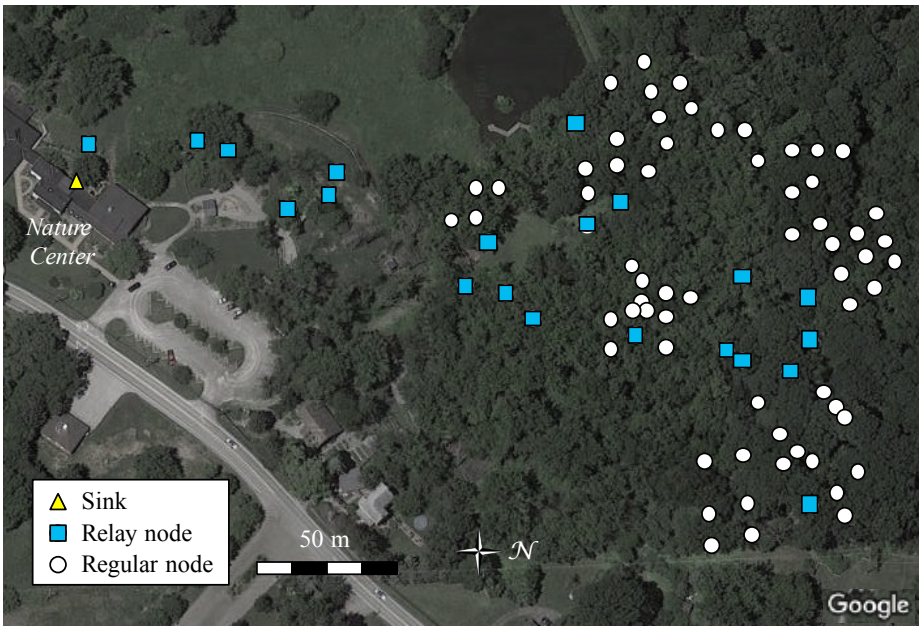


Figure 3.3. Location of the 84 WSN nodes in the ASWP WSN testbed as of August 2015.

Such location restrictions define a network topology where the sink is located in one end and the network grows in a single direction, almost in a conical shape, where the first hops have very low density and the number of nodes increases as we move farther from the sink, as seen in Figure 3.3. As mentioned above, all WSN nodes in the testbed perform sensing tasks; however, not all nodes can be placed in the locations of interest, since some nodes need to build the path from there to the sink. Nodes are classified based on their sensor configurations into two categories: *relay nodes* and *regular nodes*. Relay nodes are configured only with on-board sensors for

voltage, temperature, and humidity, and they are mainly used for connecting the locations of interest to the sink node located at the nature center. On the other hand, regular nodes are equipped with different configurations of external sensors (e.g., soil moisture, water potential, sap flow, soil temperature), in addition to the on-board sensors (i.e., voltage, temperature, and humidity), and provide the core of the environmental data. In MICAz and IRIS motes, external sensors are connected using the MDA300 data acquisition board, while TelosB motes use a custom-made board to satisfy the voltage requirements of the external sensors. From the 84 WSN nodes in the ASWP testbed, there are 21 relay nodes and 63 regular nodes, where TelosB and MICAz motes are mainly used as regular nodes, and IRIS motes are preferred as relay nodes.

Figure 3.4. Limited sun exposure in locations of interest at the ASWP testbed.

In addition, the locations of interest present very limited sun exposure for WSN nodes as seen in Figure 3.4, where the tree canopy in the forested area covers most of the field, and thus, WSN nodes in the ASWP testbed remain battery powered. Since the initial testbed deployment, different battery types and configurations have been used, aiming to balance budget limitations, space constraints (e.g., size of the enclosure), maintenance, and performance. As a result, as of August 2015, relay nodes are equipped with three D NiMH rechargeable batteries with 10,000 mAh nominal capacity, considering that they have more space available in their enclosures and

they are mostly deployed closer to the sink to forward the data from their neighbors. Regular nodes are equipped with thee AA NiMH rechargeable batteries with 2,700 mAh nominal capacity.

3.2 Preliminary Performance Analysis

During the first years the ASWP testbed operated using XMesh, Crossbow's commercial mesh networking protocol. XMesh provides a self-healing and self-organizing networking service [41]. The application code for XMesh is compiled specifically for a mote type (e.g., MICAz) and sensor board (e.g., MDA300). Arguments for programming the transmission frequency and node/group identifiers are also available. The ASWP testbed deployment uses the low power (LP) mode, where motes power off non-essential electronics when idle and still forward messages from neighbor motes. For MICAz and IRIS motes, XMesh does not support time synchronization. Therefore, all packet transmissions are made asynchronously.

XMesh's multi-hop routing is based on the Minimum Transmission (MT) cost metric aiming to minimize the total energy consumed to transmit a packet to the base station [42], similar to CTP which is based on the ETX cost metric. One of the most important differences in the design of these protocols is that XMesh employs a fixed Route Update Interval (RUI) for routing packets, while CTP adopts the Trickle algorithm to adapt the frequency of routing packets according to the network conditions, as described earlier in this chapter. Unfortunately, the source code of XMesh is not provided and then it is not possible to examine specific details of its implementation. In the ASWP testbed, XMesh was configured in LP mode with the RUI set to 128 seconds, and using an inter-packet interval (IPI) of 15 minutes.

After the summer of 2013, XMesh was replaced by an open-source approach using the original version of CTP and LPL available in TinyOS 2.1.2. The application was configured with an IPI of 15 minutes, a wakeup interval of 1 second, and a maximum Trickle interval of 60 minutes, while other parameters used default values. In addition,

the application was developed to include the necessary health and instrumentation information in order to study the performance of the protocol.

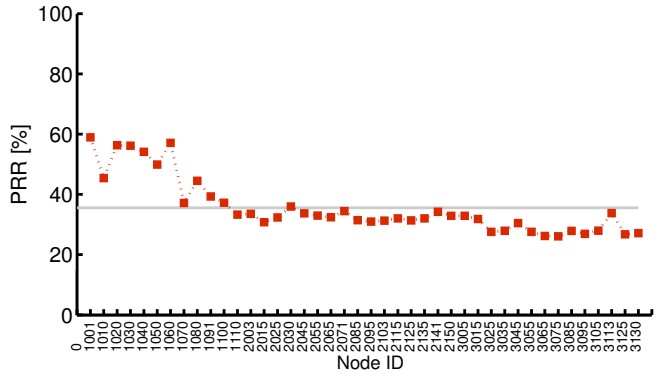
The information of the datasets used to analyze the protocols is shown in Table 3.2, and the main indicator corresponds to the *packet reception rate* (PRR), defined for each node as the ratio between packets received at the sink and the number of generated packets.

Table 3.2.
Datasets for preliminary analysis

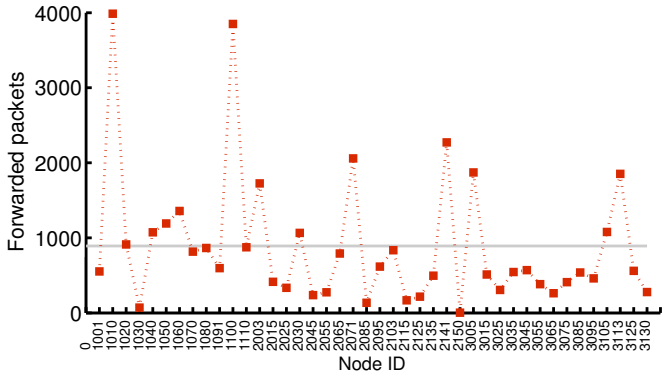
Protocol	**Time Period**	**Active Nodes**
XMesh	March 2012 - August 2012	42 nodes
CTP	November 2013 - April 2014	47 nodes

Figure 3.5 shows a summary of the results obtained from XMesh during the complete time period. The PRR for each active node and the network average are shown in Figure 3.5(a). It can be seen that overall XMesh has a low performance with a network PRR of 36.01%, where some nodes located farther from the sink can have PRRs of 27%. In addition to the low reliability in XMesh, it is observed that only a few critical nodes across the network are forwarding most of the data packets, as seen in Figure 3.5(b). In particular, multi-hop traffic is concentrated in a single node in the first two hops of the network. Data packet retransmissions are shown in Figure 3.5(c), and it can be seen that nodes with higher data traffic loads are incurring in higher retransmissions, evidencing low link quality and limitations of the routing protocol to overcome such situations. Table 3.3 shows the network average PRR of XMesh during the best month within the initial time period, reflecting that although there is a small improvement, the routing protocol still experiences the same limitations observed during the complete time period.

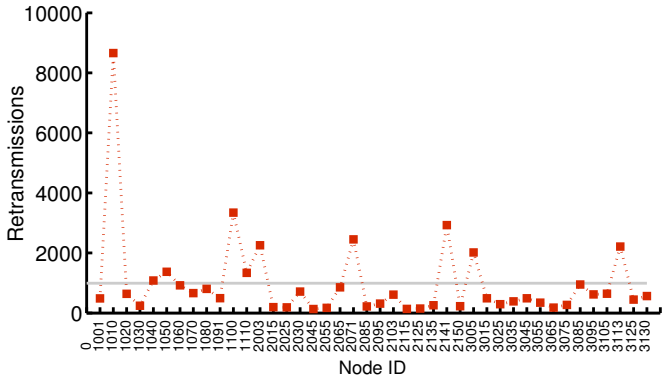
Figure 3.6 shows the results obtained from CTP. In this scenario, higher PRRs are observed for nodes closer to the sink; however, the network PRR is 79.04%, a low



(a) XMesh PRR.



(b) XMesh forwarded packets.



(c) XMesh retransmissions.

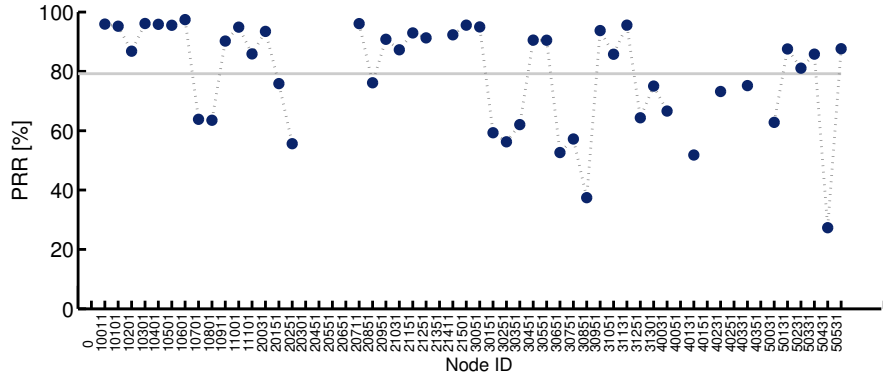
Figure 3.5. Summary of results obtained from XMesh at the ASWP testbed during the complete time period. Daily averages are shown for each node, solid lines indicate the network average, and node IDs are sorted based on their distance to the sink.

Table 3.3.
Network average PRR of XMesh and CTP

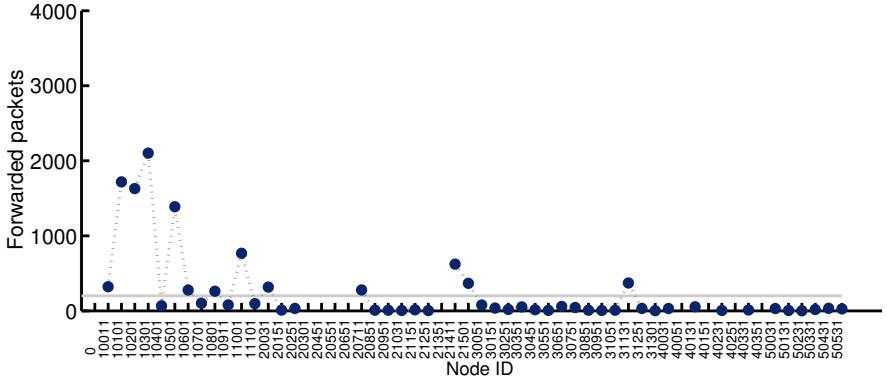
Protocol	**PRR (Complete time period)**	**PRR (Best month)**
XMesh	36.01%	42.16%
CTP	79.04%	92.71%

value for data collection applications, specially considering that there are still nodes with very low PRRs under 50%. Similar to XMesh, when using CTP, data traffic concentration is observed in critical nodes across the network, which are forwarding most of the data packets. In this case, traffic concentration is higher in critical nodes, as all other nodes forward very few or zero data packets. Figure 3.6(c) shows a more even distribution of packet retransmissions for all nodes in the testbed, critical and non-critical nodes, showing that CTP is able to correctly identify the best paths in the network. However, by concentrating the data traffic in the best routing paths, CTP is also increasing the energy consumption of those critical nodes, reducing their lifetime and creating more frequent network partitions. The effect of these network partitions is reflected in the higher PRR achieved by CTP in a shorter time period, as shown in Table 3.3.

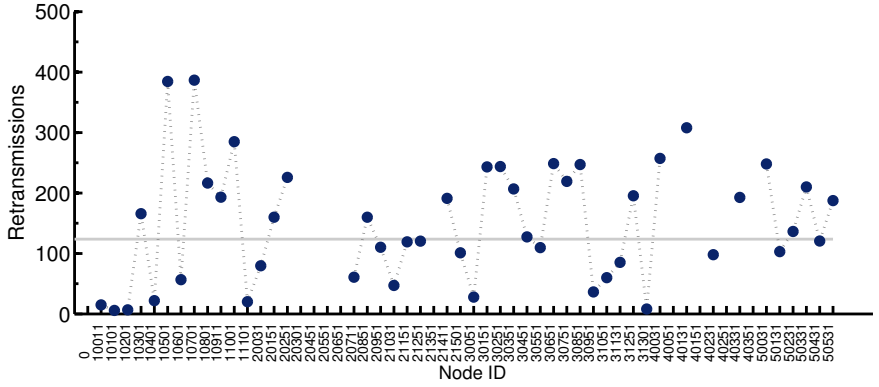
These results do not intend to present a direct comparison between the two routing protocols, but to reflect two states of the testbed with different configurations to be used as an initial performance baseline for practical WSN data collection applications.



(a) CTP PRR.



(b) CTP forwarded packets.



(c) CTP retransmissions.

Figure 3.6. Summary of results obtained from CTP at the ASWP testbed during the complete time period. Daily averages are shown for each node, solid lines indicate the network average, and node IDs are sorted based on their distance to the sink.

CHAPTER 4. ENERGY EFFICIENT AND BALANCED ROUTING (EER)

Cost-based WSN routing protocols [25] [21] have become the *de facto* standard for multi-hop data collection applications, and their principles have also been adopted by the IETF Roll working group standard RPL [43]. However, one major drawback of cost-based WSN routing protocols is that they tend to concentrate most of the data traffic on specific nodes that provide the best available routes. As a result, the energy consumption across the network is highly *unbalanced* and the busiest nodes end up depleting their batteries much faster than their neighbors, removing the best available routes first, and potentially partitioning the network.

To address this problem, we present *Energy Efficient Routing (EER)*, a new routing strategy for data collection WSNs, which exploits the WSN topology redundancy based on a controlled randomized approach without any additional routing overhead. EER, based on the concept of *parent set*, allows to select suboptimal paths in routing, reducing the data traffic load on the busiest nodes, resulting in an overall cost-effective solution that extends the network lifetime. This improvement is achieved by leveraging on the establishment of a stable routing topology, but replacing the best forwarder with a random selection from the *parent set*, defined as the subset of neighbor nodes that provide feasible routing progress towards the sink(s). Consequently, all neighbor nodes included in the parent set share the responsibility of packet forwarding, instead of a single parent node.

EER is aimed for battery-powered multi-hop WSNs for data collection, and focuses on the energy efficiency and balance achieved by the *routing layer*, which can certainly be further complemented by the energy efficiency of the MAC layer, while maintaining high reliability. Therefore, our approach can be applied to many different kinds of cost-based routing solutions, including those implemented as cross-layer optimizations to further improve their network lifetimes.

To demonstrate the proposed EER, we implement it based on the Collection Tree Protocol (CTP) [21], forming a new routing protocol called *CTP+EER*. We validate CTP+EER against the state-of-the art routing protocols CTP and Opportunistic Routing in WSNs (ORW) [44], and evaluate their reliability and energy efficiency in detail.

The specific contributions of this work are:

- We present EER, a new routing strategy that self-adapts to network conditions without the need of complicated configuration parameters, providing an energy efficient and balanced alternative for practical data collection WSN deployments. Relying on the concept of parent set, EER exploits the suboptimal network routing alternatives in WSNs, and also provides a new diagnosis mechanism that identifies nodes with strong or weak network redundancy.

- We develop CTP+EER, which extends CTP with our proposed routing strategy EER. In our implementation, the original CTP provides resource management logic and link quality estimations, while all routing logic is now controlled by EER.

- We formulate the analytical performance model for cost-based routing protocols (e.g., CTP) and their EER extensions (e.g., CTP+EER). Specifically, we provide the redundancy conditions of the network topology that guarantee CTP+EER to improve the energy efficiency at the routing layer in comparison with CTP.

4.1 Related Works

WSN routing protocols for data collection have been proposed and compared based on bandwidth utilization, reliability, latency, and energy efficiency, where CTP [21] is often used as the benchmark protocol. Protocols like BCP [45], BRE [46], and Arbutus [47] are mainly concerned about improving bandwidth utilization, increasing the

total amount of traffic supported by the network, while maintaining high reliability. These works operate on high-power conditions and thus address different scenarios than those in energy constrained data collection applications, which are the main focus of our work.

ORW [44] presents an opportunistic routing protocol for data collection applications in WSNs. The opportunistic component in ORW improves the energy efficiency of duty-cycled implementations by reducing preamble times in low power transmissions. While our work also considers multiple nodes as potential forwarders, our parent set considers link quality more strictly for possible parents and excludes nodes at the same level as the sending node, avoiding potential routing loops that affect the overall protocol performance, as we will discuss in the chapter. In addition, unlike ORWs forwarder set, we introduce an explicit construction of the parent set, enabling the examination of the topology redundancy for network diagnosis, while remaining a sender-based approach leveraging on cost-based routing mechanisms. In our work, CTP+EER is evaluated versus ORW since in both protocols the contributions of the routing layer to the total energy efficiency can be clearly differentiated from the contributions of the MAC layer.

Other works like Dozer [14] and LWB [15] have opted for cross-layer implementations, which tightly couple the behavior of routing and MAC layers. Dozer implements a basic cost-based routing protocol on top of a locally synchronized TDMA-based MAC layer. On the other hand, LWB coordinates fast network floods based on global synchronization and scheduling. Cross-layer implementations present additional challenges when they need to be implemented in multiple platforms (e.g., MICAz, TelosB, IRIS). For instance, the protocol stack needs to be re-implemented and communication parameters need to be re-configured accordingly for each new platform to replicate the desired cross-layer behaviors when using different hardware. An example would be when a WSN node from one platform requires longer time to acknowledge data packets, in which faster platforms would have to consume additional energy for idle listening in order to avoid unnecessary packet retransmissions.

EER differs from these cross-layer solutions in that it concentrates on the energy efficiency and balance achieved by the routing layer, while the main factors contributing to lower energy consumption in Dozer and LWB correspond to the MAC layer (i.e., time synchronization and scheduling). Authors of [48] present BFC, a combination of a routing protocol that removes routing packets with an adaptive LPL implementation. However, it is not clear how much contribution to the total energy efficiency in BFC comes from the MAC layer and/or routing layer. In addition, we consider that key energy efficiency factors from Dozer, LWB, and BFC are complementary to our work, since EER can be implemented on top of MAC layers that support time synchronization, scheduling, or adaptive LPL. Similarly, EER can be applied to cost-based approaches such as Dozer and BFC to further improve their network lifetimes.

Another category of related works is multipath routing, considering that with EER consecutive data packets may travel through different paths under a given WSN topology. However, the existing WSN multipath routing aims to achieve higher reliability and lower delay in data transmissions either by forwarding packets over multiple paths simultaneously, at the cost of increasing the network energy consumption [49] [50], or by using alternative paths as a backup in the event that the initial path fails [51] [52]. Our approach differs from these works because we use alternative routes as a proactive and consistent routing strategy for energy efficiency and balance, rather than reacting to a failed path event. RPL [43] defines a subset of neighbor nodes (also named a parent set) as potential parents for data collection and whenever the current best parent node fails, a new best parent node is selected from this candidate set, similar to [51] [52]. In summary, these multipath routing protocols and RPL do not focus on load balancing, incurring in higher energy consumption.

A recent approach named ORPL-LB is presented for load balancing in WSNs in [53]. It adapts the nodes' wakeup interval to control the number of potential forwarders based on an opportunistic extension of RPL. Nevertheless, ORPL-LB still has the same drawbacks of ORW because its duty cycle adaptation runs on top of

the original forwarder set which may include nodes that create routing loops. Other works on load balancing for WSNs mainly rely on one of the following methods: topology control, clustering, or adding an additional term into the routing cost function [54]. Topology control and clustering mechanisms are not directly relevant to our work, since they focus more on dense networks or require WSN nodes with special hardware components [55] [56]. Solutions that add a load balancing term into the routing cost function are proposed in [57] and [58] based on estimations of the energy available on WSN nodes, and in [47] based on the traffic processed by each node. The main drawback in these works is that defining the weight of the added load balancing term depends on each specific network scenario, and therefore, it requires a complex configuration process. Our work takes a different approach where the load balancing effect is determined by the WSN routing topology itself, without the need of additional configuration parameters. Moreover, works that rely on energy estimations must consider hardware dependent factors such as battery capacity, chemistry, age, number of charging cycles, type of sensors in WSN nodes, and environmental factors such as temperature and humidity, which introduce high variability affecting energy estimations and making this kind of methods difficult to use in practical WSN deployments.

Probabilistic approaches have been reported based on random walks [59] [60], which traded load balancing for higher energy consumption. Another probabilistic approach is presented in [61], where the routing protocol forwards packets to random nodes from the CTP routing table, following a distribution based on routing costs. However, this method has the issue of forwarding packets to the opposite of the cost gradient direction (i.e., forwarding packets to child nodes), which increases the number of hops, routing loops, and routing packets, and also affects the total energy consumption.

Finally, optimization-based approaches have also been reported [62] [63] [64]; however, most of these works introduce assumptions not practical in real scenarios (e.g., centralized computations, offline solutions, and static routing topologies), their eval-

uations are mostly based on numerical simulations, and they have not yet been tested in real implementations.

4.2 Design of EER

The design of EER has two main objectives: improve the network lifetime, defined as the time for the first node in the network to deplete its batteries, and maintain high reliability in the context of data collection applications. To achieve these goals, EER introduces the parent set concept for energy efficient and balanced WSN routing, which exploits the redundancy offered by the WSN topology diversity and reduces the traffic processed by the busiest nodes that provide the best routes in the network.

4.2.1 Energy Efficiency

The main energy-consuming components in WSN nodes are the transceiver and external sensors. In this chapter, we concentrate on the energy consumed by the transceiver, assuming that sensors have a negligible effect (e.g., low cost temperature and humidity sensors), or that other techniques are in place to manage them.

The main tasks of the transceiver affecting the network lifetime are transmissions, receptions, and idle listening. The energy consumption tradeoffs between these tasks are defined by the MAC layer, where asynchronous approaches incur in idle listening and more expensive transmissions, while synchronous approaches avoid idle listening and have short transmissions at the expense of additional control traffic overhead. Nevertheless, even at moderate data rates the total traffic load in a WSN node, which is determined by the routing layer, can be significantly increased so that the transmission task becomes the most energy consuming in the busiest nodes critical to the network lifetime.

4.2.2 Method

In general, cost-based WSN routing protocols disseminate cost information (e.g., the expected number of transmissions ETX [23]) and neighbor information carried by routing packets. EER relies on the strength of these protocols for maintaining the routing topology, while exploiting the network redundancy for energy efficiency and balance. To this end, we first propose how to measure the network redundancy, and then we show how to exploit it for energy efficiency.

For measuring the network redundancy, we introduce the concept of *parent set*, defined as a group of neighbors of a sending node that can provide *feasible routing progress* towards the sink(s). A parent set includes the primary parent node, which is the best available neighbor (i.e., the node that minimizes the routing cost of the current sending node), and additional neighbor nodes that can still provide routing progress. The parent set of a node will change dynamically throughout the node lifetime. For example, as routing costs of neighbor nodes increase over time, they may no longer be considered as members of the parent set. We note that only the information of the primary parent node is needed for establishing the routing topology, and thus the information from the other nodes in the parent set is not disseminated in routing packets.

$$LC_{xi} < LinkCost_{TH} \tag{4.1}$$

$$NC_i + LC_{xi} < NC_{PrntX} + LC_{PrntX} + 1.0 \tag{4.2}$$

$$NC_i < NC_{PrntX} + 1.0 \tag{4.3}$$

Given a node x, the feasibility of the routing progress can be defined by (4.1), (4.2), and (4.3), which are the conditions for a neighbor node i to enter the parent set of node x. In these equations, LC_{xi} represents the link cost between nodes x and

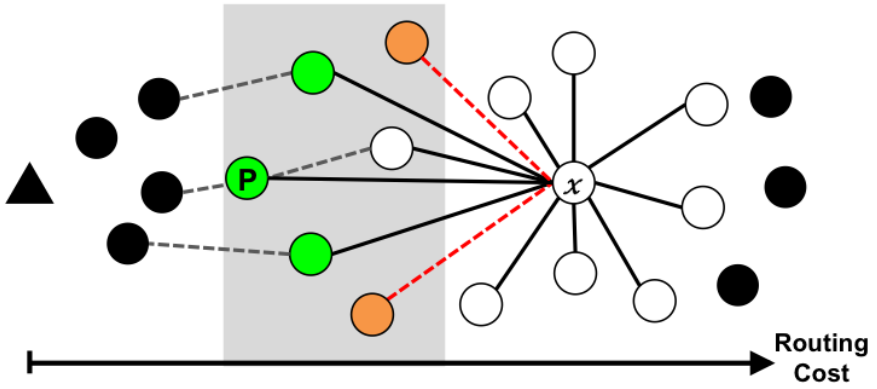


Figure 4.1. An example of the parent set of a sending node x with primary parent node P.

i; $LinkCost_{TH}$ represents the maximum link cost considered by the routing protocol; NC_i represents the routing cost of node i; NC_{PrntX} represents the routing cost of the primary parent of node x; and LC_{PrntX} represents the link cost between node x and its primary parent node. Equation (4.1) sets the maximum link cost threshold for any link to be considered by the routing protocol, determining the neighborhood of node x. Equation (4.2) specifies that a neighbor node can be considered in the parent set, only if routing progress can be made through it with appropriate link quality. In other words, if a member of the parent set is used in the route, it should not increase the total node routing cost by one perfect transmission compared to the use of the primary parent node. However, (4.2) would allow paths to be formed between members of the parent set and the primary parent node, which may occur especially when the link quality between the primary parent and the original sending node is lower compared to that of other neighbors. This situation not only increases the overall network routing cost, as data packets now travel through longer paths, but also reduces the load balancing effect of the parent set because the primary parent node is still included in the new longer path. To avoid this problem, we define (4.3) to guarantee that the route through a parent set member i will not include the primary parent node of the original sending node x. The effect of conditions defined

by (4.1), (4.2) and (4.3) is illustrated in Figure 4.1. As shown in the figure, the link cost between node x and the black nodes is higher than the maximum value defined by (4.1), and therefore, those nodes are not considered as neighbors of x. Node P corresponds to the primary parent of node x. Equation (4.2) excludes nodes with lower link quality from the parent set, as those highlighted in orange. And finally, (4.3) guarantees that the path through members of the parent set does not include the node P, qualifying the nodes (shown in green) as the members of the parent set of node x.

Once the parent set is created at each sending node, a uniform distribution is used to randomly select one of its members as the next forwarding node, ultimately distributing the data traffic across all nodes in the parent set. In this way, the parent set is controlling the use of suboptimal routes, exploiting good alternatives provided by the network topology, and thus improving the overall traffic balance. In the event that the topology does not offer any appropriate route alternatives, our method will reduce to a regular cost-based routing protocol, while still providing the network diagnosis that will be discussed later in this chapter.

4.2.3 Implementation

Architecture

In principle, the proposed EER can be effectively implemented into any cost-based routing protocol. To demonstrate, we have extended CTP [21], the *de facto* standard for multi-hop WSN data collection, to implement our proposed EER routing strategy. We refer to this implementation of the resulting new routing protocol as *CTP+EER*, where resource management logic and link quality estimation is provided by the original CTP and all routing logic is now controlled by EER.

CTP, using the ETX [23] as routing cost metric, has an architecture defined by three major components: *Link Estimator*, *Routing Engine*, and *Forwarding Engine*, as presented in Chapter 2.

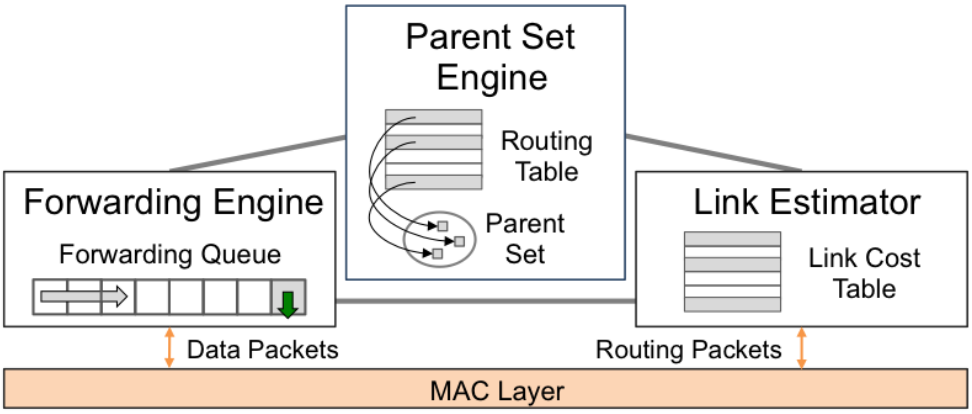


Figure 4.2. Main components of CTP+EER.

Our implementation of *CTP+EER* incorporates the Link Estimator and a modified Forwarding Engine from the original CTP. It also adds a new component named *Parent Set Engine*, which implements all routing decisions, replacing and extending the original CTP Routing Engine. The new Parent Set Engine, in addition to managing the routing table, is in charge of building and maintaining the parent set in each node, assigning the forwarding node for each data packet transmission, and defining the retransmission strategy. The architecture of CTP+EER with these three major components is shown in Figure 4.2.

To create the parent set for each sending node, the Parent Set Engine follows a stateless approach dependent upon the routing table and the link cost information provided by the Link Estimator, knowing that node routing costs and link costs already reflect historic information in their exponential smoothing filters. Whenever node routes are computed, the primary parent node is first selected, and the parent set is then formed based on conditions defined by (4.1), (4.2), and (4.3). Therefore, as node and link routing costs change over time, the parent set is recomputed without maintaining any historic information from nodes entering and leaving the set. This method reduces the memory usage of the Parent Set Engine, although it may limit some elaborated mechanisms for selecting forwarding nodes (e.g., policy based mech-

anisms). Nevertheless, we found that using a stateless approach satisfies our needs well.

Packet Retransmissions

The modified Forwarding Engine in CTP+EER handles data transmissions for packets both locally generated and received from neighbor nodes, although the Parent Set Engine now determines the strategies for routing and retransmissions. That is, the modified Forwarding Engine is mainly providing resources and logic for packet forwarding, interacting with the MAC layer, but it remains agnostic regarding the destination of the data packets and how retransmissions are decided. Routing protocols like CTP handle packet retransmissions using a single parameter that controls the maximum number of attempts, which is usually set to a high value (e.g., 30 attempts). However, in practice, a packet rarely reaches high retransmission attempts in a single hop because after each failed attempt the link cost is penalized and this will eventually trigger a parent node change. The challenge is that as the path routing cost through the current parent node increases due to link cost penalizations, the sending node may end up transmitting a data packet to neighbors initially considered as child nodes. Since this process is faster than the dissemination of node routing costs, routing loops are likely to be created. The main cause behind this problem is that the retransmission policy does not differentiate between *random errors* (i.e., link quality problems that could be overcome using retransmissions) and *bursty* or *permanent errors* that require re-routing.

In CTP+EER, we cannot use the same retransmission policy as the original CTP because it would completely ignore random errors in the links when selecting a new forwarding node from the parent set on each retransmission attempt. This would reduce the load balancing effect, as nodes with slightly better link qualities will receive most of the traffic load. In our implementation we opted to allow each sending node to have a number of retransmissions per link equivalent to the worst link quality

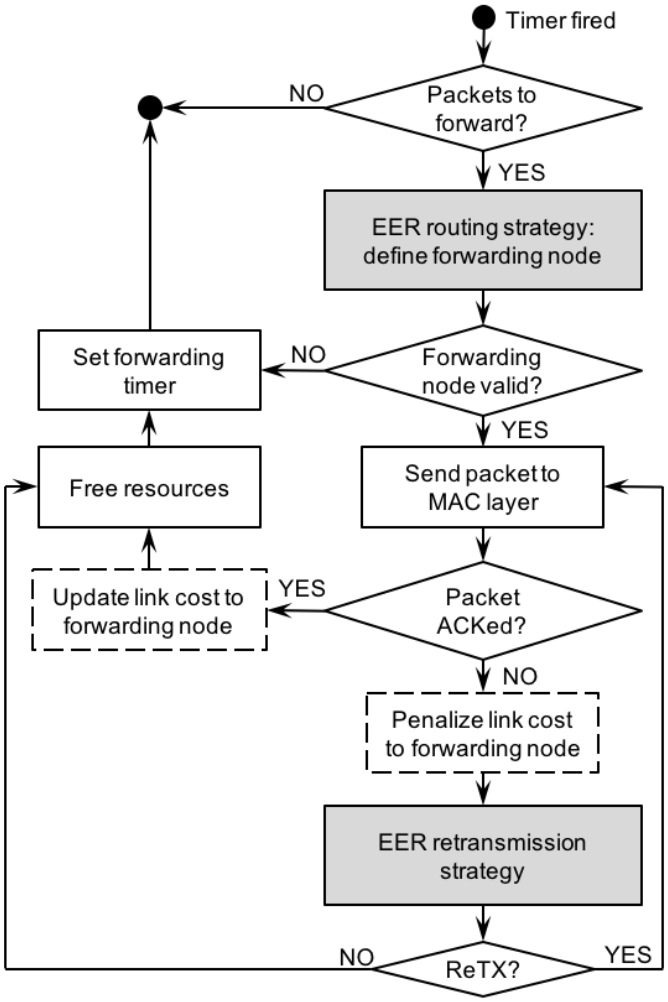
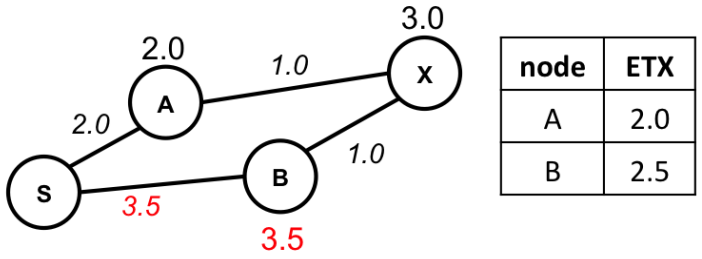


Figure 4.3. General flowchart of the packet forwarding process in CTP+EER.

considered by the routing protocol, $LinkCost_{TH}$ (e.g., 5 retransmissions, equivalent to a 20% probability of success). After such retransmissions, another member from the parent set is randomly chosen, until either the packet is successfully acknowledged or the global maximum number of retransmissions is reached. It should be noted that in our implementation, similar to CTP, retransmissions are controlled at the routing level, without using link layer retransmissions. In this way, we have a more accurate estimation of the link costs; otherwise, the routing layer only penalizes links that failed after the maximum number of link layer retransmissions. Figure 4.3 presents the



node	ETX
A	2.0
B	2.5

Figure 4.4. An example of a routing cost inconsistency in EER without relaxing the loop-detection condition.

general flowchart of the packet-forwarding process in CTP+EER, where the Parent Set Engine performs highlighted tasks and the Link Estimator updates link costs (dashed tasks).

Loop Detection

By definition, the parent set in EER does not allow routing loops to be introduced; nonetheless, delays in the dissemination of node routing costs not only affect the retransmission strategy, as discussed above, but also affect the loop detection mechanism of the Forwarding Engine. When the node routing cost of potential forwarders (i.e., members of the parent set) increases, this information takes some time to reach neighbor nodes, including the sending node. Meanwhile, the sending node will continue transmitting data packets based on its local information and setting its own node routing cost in the packet header, causing inconsistencies from the point of view of the receiving node. An example of this situation is illustrated in Figure 4.4, where the values in the network represent the link and node ETX. The table in the figure shows the local node routing costs known by node X before detecting the increment of the node ETX in B. If node X selects node B as the forwarder, when B receives a data packet from X it will detect an inconsistency and trigger new routing packets. We note that this can also occur with parent nodes in CTP; however, in CTP+EER this would be more likely to occur as the parent set size

increases, especially if we consider that in data collection applications the maximum interval for routing packets is usually larger than the inter-packet interval of data packets. To address this problem, we relaxed the loop-detection condition by adding the cost of one perfect transmission to the routing cost indicated in received packets, as defined in (4.4). This condition prevents generating unnecessary routing packets for inconsistencies detected from neighbor nodes at the same routing cost level (e.g., among members of the parent set). Then, the parent set of the node will be updated when the next routing packet is received.

$$PacketCost + 1.0 < ReceiverNodeCost \tag{4.4}$$

Our implementation of CTP+EER benefits from not requiring specific configurations other than the original parameters in CTP. In addition, EER does not introduce any new fields to the protocol header, other than the size of the parent set that may be included as instrumentation data. Finally, by randomly selecting forwarding nodes from the parent set, no changes have been applied to the routing cost function.

4.2.4 Parent Set Size for Network Diagnosis

The size of the parent set in EER provides a new indicator for network topology diagnosis. By including the size of the parent set as network instrumentation in data packets, end-users will have a better understanding of the network routing topology redundancy.

The size of the parent set ranges from one, containing only the primary parent node, up to a maximum threshold (potentially the size of the routing table). Therefore, a larger parent set reflects a node with higher routing topology redundancy, indicating that the node can distribute its traffic among multiple neighbors and also if a link failure occurs, the node still has other potential forwarders as suitable choices before attempting to re-route or being disconnected from the network.

This observation can be generalized for identifying any node with strong or weak network redundancy by examining the size of parent sets containing this node (i.e., parent set of child nodes). If all parent sets containing a given node are of size larger than one, then the node would not correspond to an articulation node (i.e., a sufficient condition), defining a *strong* node. This is a direct consequence from (4.3), which does not allow alternative routes of child nodes to go through a strong node, and therefore if the strong node fails, the children would not be disconnected from the network. On the other hand, if any child of a given node has a parent set size equal to one, the child may still be connected via re-routing if the parent node fails, defining a *weak* parent node. Still, having child nodes with parent set of size one is an undesirable situation because it reduces the load balancing effect of EER and in the event of a failure in a weak node, re-routing for locating a new path that prevents the network partition requires additional routing traffic for updating and disseminating the new node routing costs. Therefore, in practical WSN deployments, the parent set size can be used to diagnose nodes with weak network redundancy, providing new information for relocating the weak nodes or deploying additional nodes.

4.3 Analytical Performance Model

To further analyze the impact of the proposed ERR on energy consumption and network lifetime, we present an analytical model to compare the behavior of any cost-based routing protocol R (e.g., CTP) with the corresponding EER-extended routing protocol R+*EER* (e.g., CTP+EER). The connection between R+*EER* and any cost-based routing approach R is that the parent of a node in the latter will always be a member of the parent set in the former (e.g., the primary parent node). Our performance model analyzes the redundancy conditions of the network topology and their implications on practical WSN deployments.

The network model considers a WSN with N nodes represented as a destination oriented acyclic graph with adjacency matrix A, where A_{ij} represents the link from

Table 4.1.
Parameters of the analytical model of CTP+EER

Parameter	**Definition**
$\mathcal{N}$	Set of nodes in the network, where $\|\mathcal{N}\| = N$
$\mathcal{N}_{NS}$	Neighborhood of the sink
A	Adjacency matrix. A_{ij} represents the link from node i to node j
L	Link cost matrix. L_{ij} indicates the link ETX from node i to node j
d_i	Size of the parent set of node i
λ_i	Traffic processed by node i (generated and forwarded packets) in CTP
λ'_i	Traffic processed by node i (generated and forwarded packets) in CTP+EER
φ_i	Traffic generated by node i
p_i	CTP parent of node i
α	Maximum additional link cost introduced by the parent set in CTP+EER

node i to node j, with $A_{ii} = 0$. Then, each routing protocol builds its own network topology based on a given A. Note that the resulting network topology graph of R is a subset of the topology graph of *R+EER*, by the definition that the parent set in *R+EER* includes the parent node defined by R. Nevertheless, a parent node in R may not necessarily correspond to the primary parent node in *R+EER*, since multiple nodes can have the same routing costs. We focus our analytical model on the study of CTP versus CTP+EER, and the parameters of the model are defined in Table 4.1.

When comparing CTP+EER with CTP using the same configuration parameters (e.g., frequency of routing packets) and MAC layer protocol, the main improvement

on the total energy consumption will be determined by the differences in data traffic transmissions on each node (i.e., effect of the parent set in EER). Therefore, to simplify the model and reduce its complexity, we consider the energy consumed from data packet transmissions as the main factor influencing the network lifetime.

Assuming that both protocols are working under the same conditions and using the same energy sources, we define the *maximum network energy consumption (NEC)* for each routing protocol, which is inversely proportional to the network lifetime (i.e., the time for the first node to deplete its batteries). Equations (4.5) and (4.6) formulate the *NEC* of CTP and CTP+EER respectively, based on the *maximum energy consumed by each individual node* $i \in \mathcal{N}$, in worst-case scenarios.

$$NEC_{CTP} = \max_i \left\{ \left[\sum_{n=1}^{N} A_{ni}\lambda_n + \varphi_i \right] \cdot L_{ip_i} \right\} \tag{4.5}$$

$$NEC_{CTP+EER} = \max_i \left\{ \left[\sum_{n=1}^{N} A_{ni}\frac{\lambda'_n}{d_n} + \varphi_i \right] \cdot \sum_{k=1}^{N} \frac{L_{ik}}{d_i} \right\} \tag{4.6}$$

It can be seen from (4.6) that the maximum energy consumption of a sending node in CTP+EER depends on the parent set size of its child nodes and the weighted average quality of outgoing links to members of its parent set. This weighted average link quality for CTP+EER can be rewritten in terms of the link cost between the sending node and the parent node in CTP, as shown in (4.7), where $\alpha \in [1, 2)$ indicates the maximum additional link cost introduced by suboptimal nodes in the parent set of CTP+EER with respect to CTP.

$$\sum_{k=1}^{N} \frac{L_{ik}}{d_i} \geq \alpha L_{ip_i} \tag{4.7}$$

Theorem 4.3.1 *Let $d_i \geq 2$ for every $i \in \{\mathcal{N} \backslash \mathcal{N}_{NS}\}$. Then, the NEC of CTP+EER is lower than that of CTP for any multi-hop WSN.*

Proof The NEC of CTP from (4.5) corresponds to the node with the biggest subtree of descendants, such as the root of the blue subtree illustrated in Figure 4.5. In

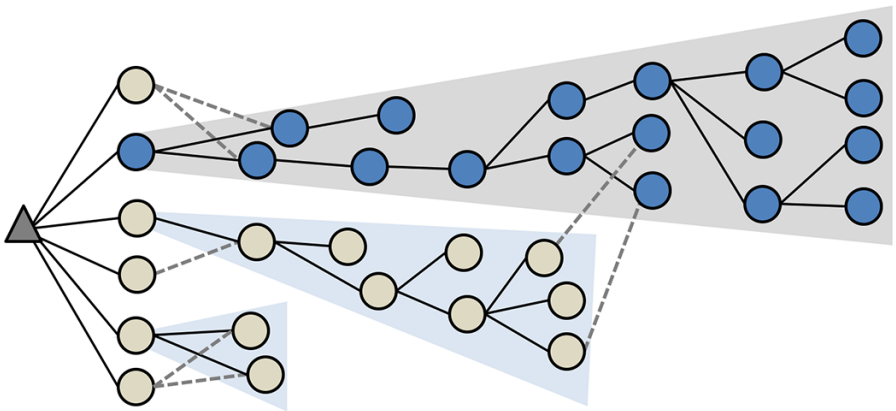


Figure 4.5. A general illustration of the subtrees of descendants where the largest blue subtree is the worst case built in CTP (solid lines) for neighbors of the sink. Dashed lines represent additional links used in CTP+EER that distribute the traffic out of the original subtree in CTP.

CTP+EER, nodes from the blue subtree can distribute a fraction of the subtree data traffic to other nodes in the network, reducing the traffic processed by the subtree root. Moreover, CTP+EER does not introduce any additional data traffic to the biggest subtree of CTP, otherwise another bigger subtree could have been defined in CTP, contradicting our initial assumption.

Assuming that $x \in \mathcal{N}_{NS}$ is the node that maximizes (4.5) in CTP, we have that

$$NEC_{CTP} = \left[\sum_{n=1}^{N} A_{nx}\lambda_n + \varphi_x \right] \cdot L_{xp_x} \, ,$$

where p_x corresponds to the sink. Then, knowing that CTP+EER does not introduce additional data traffic to the biggest subtree of x in CTP, we have that

$$\sum_{n=1}^{N} A_{nx}\lambda_n + \varphi_x \geq \sum_{n=1}^{N} A_{nx}\lambda'_n + \varphi_x.$$

Since nodes in both protocols generate the same local traffic, φ_x can be cancelled out.

In this situation, the node that maximizes the NEC from (4.6) in CTP+EER compared to CTP can either correspond to the same node, or a different node in the network.

For the first scenario, CTP+EER and CTP have the same subtree of descendants for node x, and comparing the maximum node energy consumption of x in both protocols we obtain (4.8), where the maximum node energy consumption of x in CTP+EER is equal to the NEC of the protocol. From (4.8), it can be seen that CTP+EER improves the NEC compared to CTP, for $d_n \geq 2$, proving that CTP+EER is more energy efficient than CTP in this scenario. Note that in (4.8), we have $\alpha = 1$ because x uses the same link in both CTP and CTP+EER.

$$\left[\sum_{n=1}^{N} A_{nx}\lambda_n\right] \cdot L_{xp_x} > \left[\sum_{n=1}^{N} A_{nx}\frac{\lambda'_n}{d_n}\right] \cdot L_{xp_x} \tag{4.8}$$

For the second scenario we now assume that node $y \in \mathcal{N}$ maximizes (4.6) in CTP+EER. However, knowing that x maximizes the NEC of CTP, we obtain (4.9), where the right-hand side defines an upper bound of the traffic processed by node y in CTP.

$$\left[\sum_{n=1}^{N} A_{nx}\lambda_n\right] \cdot L_{xp_x} \geq \left[\sum_{n=1}^{N} A_{ny}\lambda_n\right] \cdot L_{yp_y} \tag{4.9}$$

For y in CTP+EER we have that

$$\left[\sum_{n=1}^{N} A_{ny}\lambda_n\right] \cdot L_{yp_y} > \left[\sum_{n=1}^{N} A_{ny}\frac{\lambda'_n}{d_n}\right] \cdot \alpha L_{yp_y} \,,$$

for $d_n \geq 2$ and $\alpha \in [1, 2)$. Then, from (4.9) we have that

$$\left[\sum_{n=1}^{N} A_{nx}\lambda_n\right] \cdot L_{xp_x} > \left[\sum_{n=1}^{N} A_{ny}\frac{\lambda'_n}{d_n}\right] \cdot \alpha L_{yp_y} \,,$$

for $d_n \geq 2$ and $\alpha \in [1, 2)$. Since y defines the NEC of CTP+EER and x defines the NEC of CTP, this proves that in this scenario CTP+EER is also more energy efficient than CTP.

■

Theorem 4.3.1 indicates that if all nodes have redundancy with a parent set size greater than or equal to two, except the children of the sink(s), then CTP+EER is more energy efficient than CTP. Although the condition in Theorem 4.3.1 is strict for practical WSN deployments, we can still obtain an important observation from the theorem proof: if $\alpha = 1$, then only one child requires some level of redundancy for CTP+EER to reduce the maximum network energy consumption, NEC, with respect to CTP. Likewise, as α increases, more redundancy is needed for child nodes.

Even though we can derive the specific conditions that nodes in CTP need to satisfy for CTP+EER to be more energy efficient, we are more interested in the opposite scenario, where only the data from CTP+EER is available and we want to know whether the NEC is improved compared to CTP. In the following theorem we formalize this observation showing that only some children of the node with the highest energy consumption in CTP+EER need to provide redundancy to reduce the *maximum network energy consumption*, NEC, compared to CTP.

Theorem 4.3.2 *Let the children that concentrate $\beta/(\beta+1)$ out of the total data traffic received by the node with the highest energy consumption in CTP+EER provide redundancy with $d > \alpha$ and $\beta > d(\alpha-1)/(d-\alpha)$; then the NEC of CTP+EER is lower than that of CTP for any multi-hop WSN.*

Proof Assume that node x has M children and that it is the node with the highest energy consumption in CTP+EER (i.e., maximizes the NEC of CTP+EER). From Theorem 4.3.1, we know that

$$\sum_{m=1}^{M} \lambda_{c_m} \geq \sum_{m=1}^{M} \lambda'_{c_m} ,$$

where $c_1, \ldots, c_M$ are the children of node x. Then the upper bound of node energy consumption of x in CTP and the maximum node energy consumption of x in CTP+EER are given by

$$\left[\sum_{m=1}^{M} \lambda_{c_m}\right] \cdot L_{xp_x} \geq \left[\sum_{m=1}^{M} \frac{\lambda'_{c_m}}{d_{c_m}}\right] \cdot \alpha L_{xp_x} .$$

In the worst case, x would process the same traffic in both CTP and CTP+EER, and assuming that k of its children concentrate most of the traffic, we have

$$\lambda_{c_1} + \cdots + \lambda_{c_k} = \lambda'_{c_1} + \cdots + \lambda'_{c_k} = \beta\lambda,$$
$$\lambda_{c_{k+1}} + \cdots + \lambda_{c_M} = \lambda'_{c_{k+1}} + \cdots + \lambda'_{c_M} = \lambda.$$

If only those k children have redundancy, we can simplify the expression unifying the parent set sizes as

$$d_{c_1} = \cdots = d_{c_k} = d,$$
$$d_{c_{k+1}} = \cdots = d_{c_M} = 1.$$

Then, the energy consumption condition can be factorized as

$$[\beta\lambda + \lambda] \cdot L_{xp_x} \geq \left[\frac{\beta\lambda}{d} + \lambda\right] \cdot \alpha L_{xp_x},$$

and we can solve for β.

$$\beta > \frac{\alpha L_{xp_x} - L_{xp_x}}{L_{xp_x} - \frac{\alpha L_{xp_x}}{d}} = \frac{d(\alpha - 1)}{d - \alpha}, \textit{ for } d > \alpha \tag{4.10}$$

From (4.10) it can be seen that if the k children that concentrate $\beta/(\beta + 1)$ out of the total received traffic of node x in CTP+EER provide redundancy with $d > \alpha$, its maximum node energy consumption in CTP+EER would be lower than its corresponding node energy consumption in CTP. Note that if $x \in \mathcal{N}_{NS}$, then $\alpha = 1$, since both protocols would be using the same link, and therefore only one child of x would need to provide some redundancy with $d > 1$.

Knowing that x maximizes the NEC in CTP+EER and that its corresponding node energy consumption in CTP would be higher proves that the NEC of CTP+EER is lower than that of CTP.

■

From Theorem 4.3.1 and Theorem 4.3.2, we can see that the redundancy conditions are highly dependent on the value of α from (4.7). For example, when comparing the nodes node that maximizes (4.6) in CTP+EER, if $\alpha = 1.3$, according to Theorem 4.3.2 only those child nodes that concentrate 50% of the node traffic in CTP+EER need to have network redundancy with $d \geq 2$ for CTP+EER to reduce the NEC compared to CTP. Other child nodes that concentrate the remaining 50% of the node traffic would not be required to have network redundancy in CTP+EER. We note that for scenarios where the performance data from CTP is not available, α could be estimated comparing the link quality of primary parents to that of other members of the parent set, considering that primary parent nodes or nodes with similar link quality would be used as parents in CTP.

We can also observe that when using the parent set for diagnosis of network redundancy as defined earlier in the chapter, nodes with very weak network redundancy (i.e., nodes only within the parent sets of size one) indeed may not satisfy Theorem 4.3.2. Then, as weak nodes are identified, their network redundancy can be addressed (e.g., node relocations, deploying additional redundancy nodes) proactively; hence the diagnosis from the parent set can lead the network to satisfy the requirements of Theorem 4.3.2. In the next chapter we will discuss practical values of α in more detail, and how Theorem 4.3.2 can be applied to the analysis of practical experiments of CTP+EER.

CHAPTER 5. EVALUATION OF EER

In this chapter, we evaluate the CTP+EER protocol and demonstrate its significance in comparison with CTP and ORW via testbed experiments and simulations. Then, we present a case study of the deployment of CTP+EER in the ASWP testbed, for a duration of 31 days with more than 160,000 collected packets, to evaluate the protocol and its diagnosis functionalities applied to the testbed regular operation.

5.1 Experiments and Simulations

To validate our CTP+EER routing protocol, we performed a series of WSN experiments and simulations developed in TinyOS 2.1.2, and compared the results of CTP+EER with those obtained by CTP and ORW, two state-of-the-art approaches using traditional cost-based and opportunistic routing strategies, respectively. WSN experiments were conducted in the publicly available Indriya testbed [34] deployed at the National University of Singapore, using 95 TelosB motes accessible at the time of the experiments (between January and August 2015). Further evaluations were conducted using Cooja [65] to emulate the same TinyOS applications compiled for TelosB motes and used for the testbed experiments.

The evaluation is based on the following metrics:

- *Packet Reception Rate (PRR):* defined for each node as the ratio between the number of data packets received at the sink and the number of generated packets.

- *Transmission Cost:* defined for each node as the ratio between the total number of data packet transmissions (i.e., generated, forwarded, and retransmitted packets) and the number of generated packets.

- *Duty Cycle:* defined for each node as the percentage of time the radio is active.

The three routing protocols are evaluated using a WSN application with an average inter-packet interval (IPI) of 4 minutes, a reasonable value for requirements in data collection applications with low rates. All protocols also use the same LPL implementation based on the CC2420 driver included in TinyOS, although ORW introduces some modifications as discussed in [44]. CTP+EER and CTP are configured with an LPL wakeup interval of 1 second, while ORW is configured with 2 seconds (denoted as *ORW(2s)*), which resulted in the optimal configuration for each protocol in our tests. We also repeated the experiments with ORW using the LPL wakeup interval configuration of 1 second (denoted as *ORW(1s)*), and discuss its effects on the duty cycle. The sink node in all the experiment scenarios is awake 100% of the time.

For a fair comparison, we examined other default parameters of the LPL implementation for TelosB motes in TinyOS. We found that the default time interval for listening after an LPL sleep interval is actually not large enough to detect a data packet transmission and to receive the corresponding acknowledgement. To achieve fast data packet acknowledgements, CTP+EER and CTP were configured to use TinyOS hardware acknowledgments, whereas ORW continued using the default TinyOS software acknowledgments due to the unavailability of hardware acknowledgments in its implementation. The LPL listening time is controlled by the maximum number of Clear Channel Assessment (CCA) checks done by the CC2420 driver, which defines a default value of 400. For this value, we found that basic packet transmissions over a single hop are correctly acknowledged only around 60% to 70% of the time, depending on the data packet size, introducing unnecessary packet retransmissions and increasing the energy consumption. In our validation, we use data packets with a payload size of 60 Bytes, and we found that using 1100 maximum CCA checks resulted in 100% acknowledged packets for transmissions between TelosB motes in scenarios with low interference, a configuration that was used for the three routing protocols in our tests.

At the routing level, CTP+EER and CTP use a maximum Trickle timer interval of 30 minutes for routing packets and a maximum of 10 retransmission attempts for data packets, while ORW uses its default parameters. For our tests with CTP+EER, we also defined a maximum parent set size of 5 neighbor nodes.

5.1.1 Experiments in Indriya

The 95 TelosB motes used in Indriya are distributed among three floors, and we chose node one as the sink, which is located in a corner of the first floor. Our tests are based on average values of 2-hour runs repeated at least 4 times, for each routing protocol.

We start by characterizing the WSN topology in Indriya and we use the average hop counts obtained by CTP, which always uses the best available neighbor to forward data packets, and provides an accurate distribution of nodes by hop count in the testbed. As shown in Figure 5.1, WSN nodes in Indriya are heavily concentrated close to the sink node, where the farthest nodes are located 5 hops away, but 67% of the nodes are within 3 hops of the sink. We use these average hop counts obtained from CTP to sort the nodes based on their path distance in the discussion of the following results.

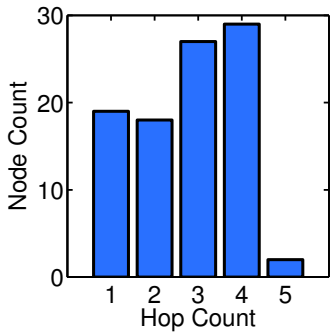


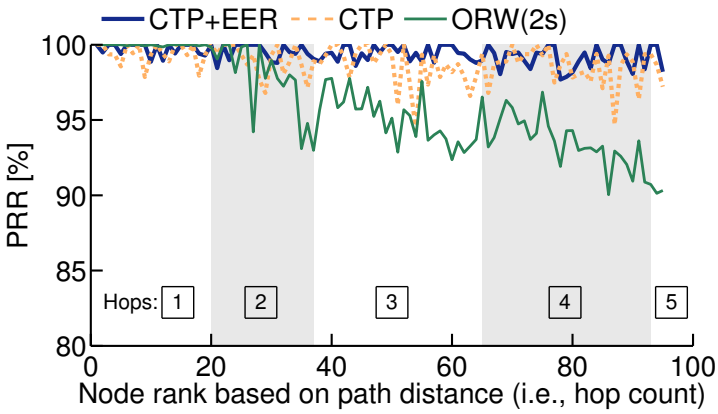
Figure 5.1. Path length distribution in Indriya based on CTP.

Table 5.1.
Average performance of CTP+EER versus CTP and ORW in Indriya

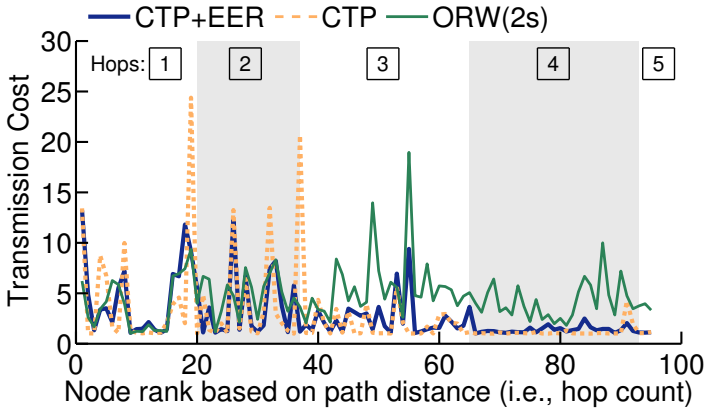
Indicator		**CTP+EER**	**CTP**	**ORW(2s)**	**ORW(1s)**
PRR [%]	**Avg.±Std.**	99.41±0.61	98.81±1.17	95.99±3.04	95.68±3.03
	Min.	97.68	94.30	90.03	80.59
	Max.	100	100	100	100
TX Cost	**Avg.±Std.**	2.77±2.66	2.68±3.92	4.66±2.66	2.99±2.31
	Min.	1.01	1.00	1.02	1.06
	Max.	13.20	24.36	18.96	15.79
Duty Cycle [%]	**Avg.±Std.**	3.27±0.69	3.20±0.92	1.62±0.79	4.13±2.02
	Min.	2.36	2.36	0.34	1.69
	Max.	6.15	7.81	5.97	6.93

Reliability

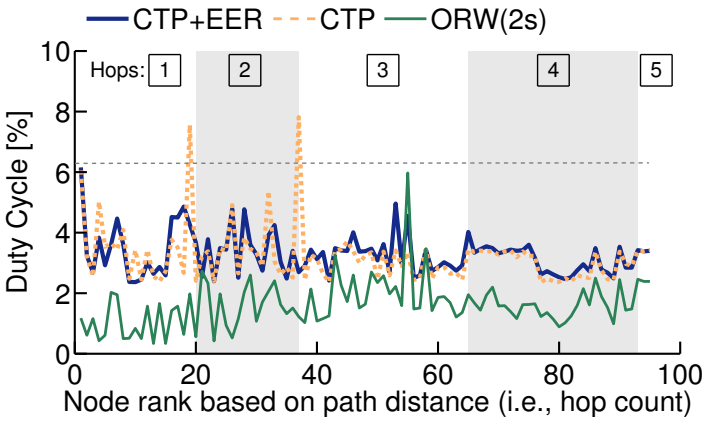
We compare the reliability of the routing protocols based on their PRR. The results obtained are summarized in Table 5.1, showing that CTP+EER has the highest average node PRR, with lowest standard deviation, among the three routing protocols in our experiments. Detailed results for each node are shown in Figure 5.2(a) with nodes sorted based on their path distance. It can be seen that overall, nodes with CTP+EER achieve the highest PRR without drastic fluctuations as observed for CTP and ORW. One observation for ORW is that the node PRR consistently decreases as the node path distance to the sink increases. As a result, nodes located 4 or 5 hops away can have up to a 10% lower PRR in ORW compared to CTP+EER. The parent set in CTP+EER explicitly addresses this problem by providing routing progress after each transmission, as discussed in Chapter 4. The effectiveness of CTP+EER is shown by the average PRR higher than 99%, evidencing high network reliability.



(a) PRR



(b) Transmission cost



(c) Duty cycle

Figure 5.2. Results from experiments in Indriya.

Energy Efficiency and Balance

We evaluate the energy efficiency and balance of the routing protocols based on the transmission cost and duty cycle. The transmission cost provides a routing level indicator of the traffic load on each node. On the other hand, the duty cycle provides a direct metric on energy consumption, which includes the effects from both routing and MAC layers. Also, the node with the maximum duty cycle corresponds to the node that determines the network lifetime. These results need to be interpreted based on the PRR achieved by each routing protocol, since lower PRRs may reduce the data traffic load processed by the busiest nodes in the network, influencing their transmission cost and duty cycle.

Figure 5.2(b) shows the average node transmission cost for the three routing protocols. It can be seen that CTP has nodes with the highest transmission cost, corresponding to the busiest nodes in the first two hops of the network topology. In contrast, CTP+EER and ORW do better in distributing the traffic load, especially for nodes within two hops from the sink. In particular, as shown in Table 5.1, CTP+EER is able to reduce the maximum transmission cost by 45% and 30%, compared to CTP and ORW(2s), respectively. These results, together with the lowest average transmission cost, reveal how unbalanced the traffic is for WSN nodes with CTP. The experiments show that CTP+EER is able to improve the energy efficiency at the routing layer, while achieving the highest average PRR among all the three tested routing protocols.

As shown in Figure 5.2(b), the node with the maximum transmission cost in ORW(2s) is located 3 hops away from the sink, which reflects that ORW is experiencing looping packets. These looping packets are dropped when detected, causing the lower PRR observed in Figure 5.2(a).

Regarding the results of duty cycles, as shown in Figure 5.2(c) and Table 5.1, nodes in CTP reach the highest duty cycle of 7.81%, whereas CTP+ERR achieves a maximum duty cycle 21% lower than CTP, and only 3% higher than ORW(2s). Note

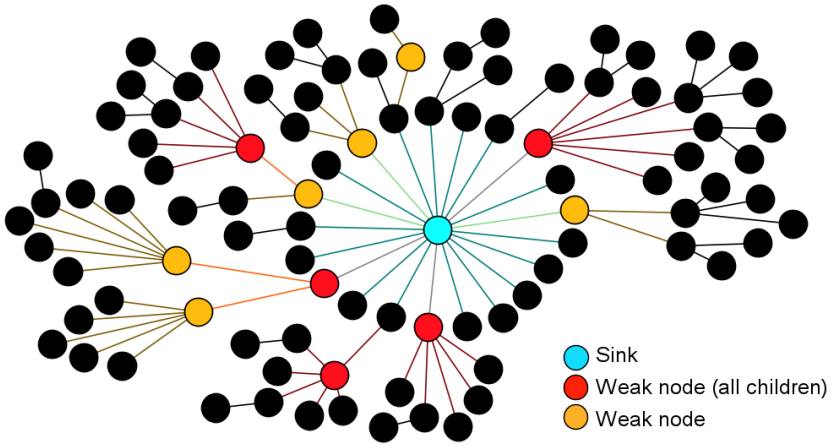


Figure 5.3. Representation of the network topology of experiments using CTP+EER in Indriya.

that when using the same LPL wakeup interval of 1 second, CTP+EER actually achieves a maximum duty cycle 11% lower than ORW(1s).

Network Redundancy

We examine the network topology redundancy offered by Indriya using the size of the parent set as an indicator. Figure 5.3 shows a representation of the network topology when using CTP+EER, where the sink node is shown in the middle of the diagram, and edges connect nodes to their most frequent forwarder. Nodes are highlighted based on their diagnosis, where black nodes correspond to nodes with strong network redundancy, and weak nodes are highlighted in red or yellow. Red nodes are the ones that in the case of failure would at least temporarily disconnect all their children (i.e., all their children have an average parent set of size one), while yellow nodes are the ones that would disconnect at least one child.

The diagnosis of the topology in Indriya via CTP+EER discovered 5 nodes with the lowest level of redundancy (i.e., red nodes). These nodes are receiving most of the traffic from 23 direct children and 24 extended children (i.e., children of direct chil-

dren), covering traffic from about 50% of the nodes in the network. With additional 6 yellow weak nodes, there are a total of 11 weak nodes at the risk of partitioning the network. This weak network redundancy would certainly exist with CTP, but would not be identified before the network is finally partitioned due to the failure of one of the red or yellow nodes from Figure 5.3.

In a practical WSN deployment, once these weak nodes are identified, their locations can be analyzed to determine if they can be relocated or if additional WSN nodes can be deployed to provide new alternative paths towards the sink.

Performance Model Verification

The experiment results also allow us to examine the network topology observed in Indriya based on our analytical model introduced in Chapter 4. From the link quality of the nodes in CTP+EER and CTP, we found that α has an average of 1.01 ± 0.03, with a maximum value of 1.30. Given this maximum value of α, CTP+EER can improve the network energy efficiency compared to CTP when the child nodes responsible for 50% of the data traffic processed by the node with the highest energy consumption have a parent set size greater or equal than two in CTP+EER. This condition is in fact satisfied for all the black nodes in Figure 5.3, and was confirmed for the busiest node in CTP+EER, which corresponds to one of the yellow nodes.

5.1.2 Simulations in Cooja

Knowing the limitations of the network topologies in WSN testbeds available to the community, we further conduct our validation of CTP+EER using the Cooja simulator. Our simulations use the Unit Disk Graph Medium (UDGM) with exponential distance loss as radio model and a maximum link quality of 90% to account for uniform random noise during packet transmissions. The assumptions in this radio model are idealistic, but our main objective with the simulations is to evaluate the three routing protocols under different topologies, based on the results observed from

Table 5.2.
Summary of results from simulations with 20 WSN nodes

Indicator		CTP+EER	CTP	ORW(2s)
PRR [%]	**Avg.±Std.**	99.97±0.07	99.96±0.09	86.36±7.35
	Min.	99.74	99.73	76.98
	Max.	100	100	100
TX Cost	**Avg.±Std.**	4.11±2.94	4.17±5.71	7.06±2.01
	Min.	1.01	1.01	2.43
	Max.	8.96	16.90	10.07
Duty Cycle [%]	**Avg.±Std.**	2.75±0.52	2.72±0.86	2.65±0.93
	Min.	2.20	2.12	1.04
	Max.	3.77	4.98	4.25

the above testbed experiments. Our simulations in Cooja ran 24 hours of the WSN application.

Effect of the Network Topology

Our experiments in Indriya captured a behavior in ORW, which reduces the PRR for nodes with a larger path distance from the sink. To further investigate this situation, we started using a simple rectangular topology of 20 WSN nodes distributed along 7 hops with three nodes in each hop level and one node in the last level. The three routing protocols were simulated in this topology and the summary of the results is shown in Table 5.2.

As expected, CTP+EER and CTP achieve similar PRRs above 99%. CTP+EER also reduces the maximum transmission cost by 46% and the maximum duty cycle of the busiest nodes that define the network lifetime by 24%, compared to results obtained by CTP. Again, ORW(2s) suffers from additional packet drops, reducing its average node PRR to 86.36%. CTP+EER still improves the maximum transmission

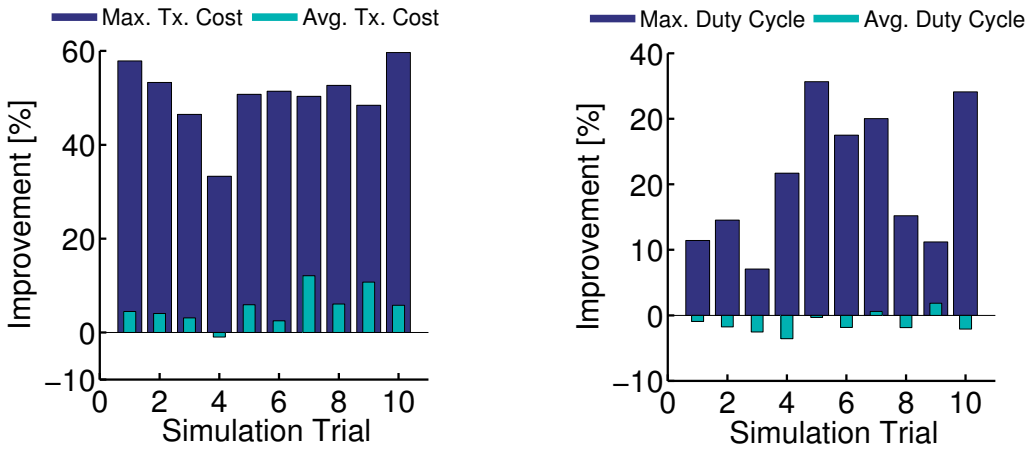
cost and maximum duty cycle by 11% compared to ORW(2s). Note that the PRR of ORW(2s) further decreased compared to that obtained in the Indriya testbed, due to the more even distribution of nodes across the different hop levels. When routing loops occur, a higher percentage of the nodes in the network would be affected, unlike the Indriya testbed where nodes are heavily concentrated close to the sink. Similar to the results obtained from Indriya, the lower PRR of ORW(2s) still affects the results for the transmission costs and duty cycles, where nodes closer to the sink process less traffic due to packet drops, but nodes involved in the routing loops increase their energy consumption. This confirms that ORW reduces the performance of nodes located farther from the sink, depending on the network topology and the distribution of the nodes in the network.

Random Network Topologies

We also conducted our evaluation using random network topologies of 100 WSN nodes. For these scenarios, the first 2 hops of the networks are fixed with 4 and 5 nodes, respectively, where nodes in the second hop can communicate with at least 2 nodes from the first hop. The remaining nodes are uniformly and randomly distributed in an area of 350x350 m^2, where WSN nodes have a maximum transmission range of 50m, and the sink is located in one of the corners. Controlling the first two hops of the network guarantees that nodes are not heavily concentrated around the sink and also creates potential critical nodes with a minimum network redundancy.

Considering the lower PRR performance of ORW in our previous experiments and simulations, we now focus on the improvement of CTP+EER in comparison with CTP. In this scenario, simulations are repeated for 10 random topologies, which results in networks with 10 hops in diameter, with up to 15 nodes in each hop level.

For these simulations, CTP+EER and CTP achieve an average PRR of 99.88% $\pm$ 0.02% and 99.93% $\pm$ 0.01%, respectively. In all simulation trials, both routing



(a) Transmission cost

(b) Duty cycle

Figure 5.4. Improvement of the transmission cost and duty cycle in CTP+EER compared to CTP in simulations with random topologies in Cooja.

protocols maintain node PRRs higher than 99%, showing that they have no problems processing the traffic load in the network under the assumptions of the radio model.

As shown in Figure 5.4(a), CTP+EER reduces the maximum transmission cost in all simulation trials compared to CTP, with the improvements ranging from 33.29% to 59.66%. The improvements of CTP+EER in the maximum duty cycle are shown in Figure 5.4(b), achieving reductions between 7.06% and 35.67%. The load balancing effect of the parent set in CTP+EER can be clearly seen in the figures, greatly reducing the energy consumption in the busiest nodes with CTP and thus increasing the network lifetime.

5.1.3 Discussion

Our evaluation results show that overall CTP+EER achieves and maintains high reliability, with average PPRs above 99%, in both testbed experiments and simulations. CTP has similar PRR results in the simulations but a slightly lower performance in the testbed. The link diversity introduced by the parent set in CTP+EER

allowed WSN nodes to explore additional paths reducing the number of packet drops. Overall, CTP+EER improves the energy efficiency at the routing layer compared to CTP by reducing the maximum transmission costs, which is observed in the testbed and in all simulations. The energy efficiency of the routing layer in CTP+EER results in reductions of the maximum duty cycle ranging from 7% up to 35% compared to CTP, extending the network lifetime.

ORW presented a different behavior in the testbed, where nodes located far from the base station have PRRs up to 10% lower than the same nodes in CTP+EER. This is confirmed in simulations using a topology with WSN nodes more evenly distributed across multiple hops. In these scenarios, the lower PRR in ORW is mainly caused by packets looping between nodes with similar routing costs (i.e., EDC in ORW), which are dropped when detected.

In comparison with the optimal ORW(2s) configuration, CTP+EER reduces the maximum transmission costs about 30% and was only 3% higher for the maximum duty cycle in the testbed, when ORW(2s) runs on a different MAC layer configuration that saved close to half of the energy CTP+EER consumed in LPL idle listening. The improvement of the energy efficiency at the routing layer of CTP+EER compared to ORW is confirmed by the 11% reduction of the maximum duty cycle compared to ORW(1s) in the testbed experiments, when both routing protocols were using the same LPL wakeup intervals of the MAC layer configuration. Moreover, in the simulation with a larger network diameter and nodes more evenly distributed across the different hop levels, CTP+EER reduces both maximum transmission cost and the maximum duty cycle by about 11%, when compared to ORW(2s).

We note that the improvement in the maximum transmission cost (i.e., energy efficiency at the routing layer) indicates the potential improvement in maximum duty cycle (i.e., network lifetime), where the duty cycle captures the effect in energy consumption from both routing and MAC layers.

Finally, the analysis of the parent set size for different nodes has shown that even though the WSN topology may present nodes with weak network redundancy and

high network traffic, CTP+EER would still be able to improve the energy efficiency compared to CTP and ORW. For example, while the topology in Indriya has multiple weak nodes close to the sink in our tests, CTP+EER is able to meet the worst-case redundancy requirements derived from our analytical performance model and therefore is also able to improve the network energy efficiency.

5.2 Case Study: ASWP WSN Testbed

We present the case study of the deployment of our proposed CTP+EER routing protocol in the ASWP WSN testbed introduced in Chapter 2.

5.2.1 WSN Application Description

For the ASWP testbed we use the same TinyOS application described in the previous section for CTP+EER, only changing the IPI of data packets to 30 minutes and the maximum Trickle timer interval to 40 minutes, which satisfy the requirements of our environmental monitoring application. In addition, the WSN application incorporates the corresponding drivers for on-board and external sensors as needed, depending on their specific sensor configuration.

The data packet structures have also been modified accordingly to include only the necessary sensor and instrumentation data for evaluating the protocol performance, including the size of the parent set, primary parent, and current parent node. For the ASWP testbed, data packets have a payload size of 57 Bytes.

5.2.2 Protocol Evaluation

For this case study, we selected a dataset with more than 160,000 packets that correspond to all collected data during 31 days from August 15 2015 to September 14 2015, where nodes performed over 1.4 million data packet transmissions. Before this time period, the testbed was operating regularly and battery replacements were

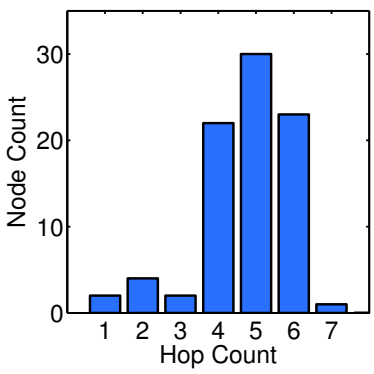
Table 5.3.
Summary of results for CTP+EER from the ASWP WSN testbed

CTP+EER	**Avg. ± Std.**	**Min.**	**Max.**
PRR [%]	99.35 ± 1.21	91.53	100
TX Cost	7.03 ± 9.11	1.12	66.56
Duty Cycle [%]	2.61 ± 0.67	1.58	5.10

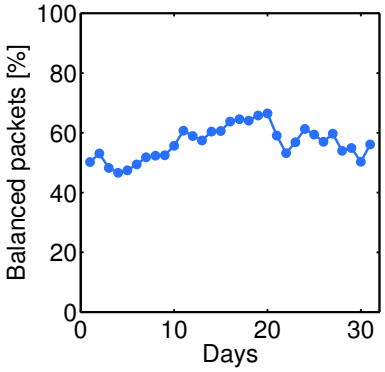
determined by periodic maintenance operations; therefore, nodes did not have fresh batteries at the start day of the case study period. In this day, the network was re-established after a WSN application update was deployed in the testbed with all 84 active nodes. During the time period covered by the dataset 10 nodes depleted their batteries, introducing additional network dynamics.

We start the evaluation by computing the path length distribution of CTP+EER in the ASWP testbed, as shown in Figure 5.5(a). Even though the distribution in Indriya was previously computed using CTP (see Figure 5.1), it can be seen that both networks have very different topologies. Unlike the topology in Indriya, nodes in the ASWP testbed are concentrated in distant hops from the sink due to location restrictions, presenting a real scenario where load balancing is critical and nodes in the first few hops are going to concentrate all the data traffic in the network, not only the fraction of the total data traffic generated by their descendants. Moreover, redundancy is limited in the first few hops and then the routing protocol needs to really exploit the few available options.

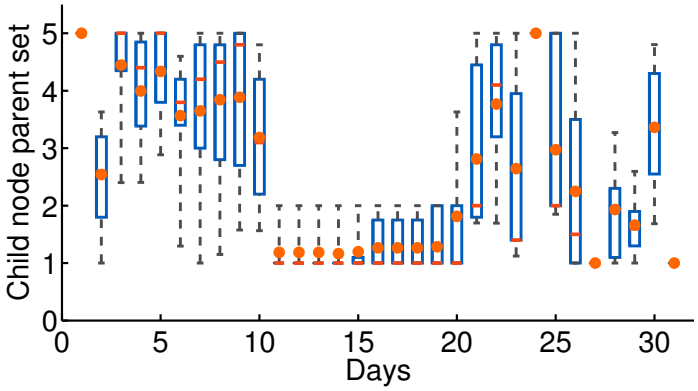
The performance of the protocol is evaluated using the same indicators as in the previous experiments, and the results are shown in Table 5.3. Nodes in the ASWP testbed using CTP+EER achieve similar PRR results compared to the experiments in Indriya, but with higher standard deviation, which is expected due to the additional dynamics introduced by the larger time period of the dataset and nodes depleting their batteries.



(a) Path length distribution based on the primary parents of CTP+EER



(b) Percentage of data packets using alternative paths



(c) Daily boxplots of the size of parent sets containing the node with ID 12 (Blue boxes indicate 25th and 75th percentiles, red bars indicate the median, and orange dots indicate the mean)

Figure 5.5. Results from the evaluation of CTP+EER in the ASWP testbed.

The data traffic concentration caused by the network topology in ASWP is reflected in the maximum transmission cost, indicating that the busiest node is processing a traffic load higher than 66 times its own generated traffic (i.e., approximately the traffic load generated by 66 nodes). This node in ASWP achieves a duty cycle of 5.10% after the 31 days as a result of the higher traffic load.

Network Diagnosis

When analyzing a testbed deployment for regular operation, different from a benchmark experiment, it is necessary to establish how different parameters are changing over time. In Figure 5.5(b), we show the daily percentage of data packets that were forwarded using alternative paths, ranging from 46% to 66%. This shows the data traffic load that is removed from the primary parent and is now forwarded by a different member of the parent set.

The same analysis can be used to monitor weak and strong nodes in a WSN deployment over time. In the ASWP testbed, the busiest node (i.e., the node with the highest transmission cost and duty cycle) is the node with ID 12, and we use this node to show the behavior of the parent set size of child nodes in multiple days. Figure 5.5(c) has a daily boxplot for the size of the parent sets containing node 12 (i.e., parent sets of child nodes). During the first day, while the network is establishing, it can be seen that children of node 12 report the highest level of redundancy, but different patterns are observed later on. In general, node 12 is 2 or 3 hops from the base station and it has up to 34 different children with redundancy that changes as a result of updates in the network topology. In particular, two nodes that were providing redundancy to the children of node 12 depleted their batteries on day 10 and 20. After day 10, the redundancy reported by children of node 12 is reduced, but it increases after day 20 when node 12 tends to prefer paths of 3 hops. After these topology updates, child nodes see more redundancy when node 12 uses longer paths, but for shorter paths, children end up reporting that no other alternative is available, as seen in days 27 and 31. The selection of shorter or longer paths in node 12 depends only on the node and link ETX reported by their neighbors.

While weak and strong nodes depend on the current state of the network routing topology, it is CTP+EER that enables to monitor and diagnose this behavior over time and this functionality can be integrated into and exploited by the network management system. After diagnosing the nodes in ASWP, we did not find any weak

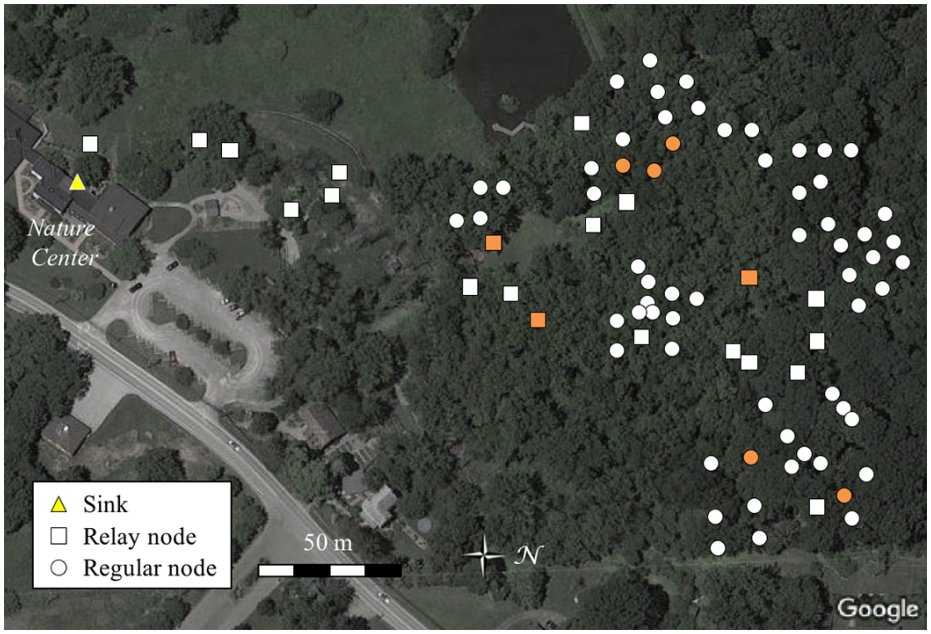


Figure 5.6. Location of the 84 WSN nodes in the ASWP WSN testbed as of August 2015. Weak nodes diagnosed by CTP+EER are highlighted in orange.

node consistently reporting that all its children had a parent set of size one for each day of the study (unlike red nodes in Indriya from Figure 5.3); although nodes like 12 did report this weakness towards the end of the study. Additional weak nodes were identified where at least one child has a parent set of size one and therefore could potentially be disconnected in the event that the parent node deplete its batteries or fails. Nodes that reported this behavior in the first days of the study are highlighted in Figure 5.6, where node 12 is the closest highlighted node to the sink.

Addressing the weakness in these nodes depends on different factors. For example, node 12 with up to 34 children will be diagnosed as weak if at least one child reports no alternative paths; therefore, the priority of addressing this weakness may depend on the importance of the specific child node. This can be determined based on the traffic load the child is processing or the specific data being measured by its sensors.

Performance Model Verification

Even though in this case study at the ASWP testbed we did not have the CTP test for this network topology and configuration, the parameters of the performance model derived in Chapter 4 can be approximated based on the data collected from primary parent nodes in CTP+EER, considering that nodes with similar link quality would be selected as parent nodes in the original implementation of CTP. Based on the first days of the collected data when all nodes were active, and using the link quality of the primary parent nodes, we found that α has an average of 1.03 ± 0.05, with a maximum of 1.33.

To satisfy the requirements of Theorem 4.3.2, the child nodes that concentrate 50% of the data traffic processed by node 12 must have a parent set size greater or equal than two. From Figure 5.5(c), it can be seen that most children of node 12 satisfied this condition during the time all nodes were active, showing that CTP+EER improves the network lifetime compared to CTP in the ASWP testbed. As the network redundancy was reduced, the network lifetime achieved by CTP+EER decreased, while the risk of a network partition also increased. Still, this situation can be diagnosed by CTP+EER indicating that network maintenance is required.

CHAPTER 6. ENERGY PROFILES

As described in Chapter 4, the energy efficiency of a WSN application mainly depends on tasks related to the transceiver and external sensors, where energy consumption tradeoffs between tasks associated to the transceiver are defined by the MAC layer. We also showed how our proposed EER routing strategy balances the data traffic in the network, reducing data transmissions on critical nodes and improving the network lifetime.

In this chapter, we now study the effect of the MAC layer on the network energy efficiency by analyzing the energy consumption profile of WSN nodes, including the effect of external sensors. These energy profiles combine health and instrumentation information received from network deployments with laboratory measurements of the energy consumed by each individual task of the WSN application. We compute the energy profile of WSN nodes deployed at the ASWP testbed and compare them with laboratory experiments to provide an additional insight into the network dynamics and changes in energy consumption. Then, we present an estimate of the node lifetime and discuss the uncertainty of these estimations.

6.1 Related Work

Existing studies in this area conduct the energy consumption analysis from the viewpoint of WSN communications. However, practical WSN deployments include multiple sensors for each node and this sensing activity affects the nodes energy consumption.

TinyOS [4] provides a development environment that integrates TOSSIM [66], a tool for application simulation. TOSSIM replaces components at different levels of the application for simulation implementations and offers an efficient alternative

for evaluating high-level applications. However, these simulations have significant limitations for energy consumption analysis, since hardware details have been removed from the application.

There have been different efforts for integrating detailed hardware and energy models into simulation tools to obtain more accurate energy consumption estimations. PowerTOSSIM [67] presents an extension to the TinyOS simulator integrating an energy consumption model for Mica2 motes. AEON [68] presents an evaluation tool to quantitatively predict the energy consumption of a WSN mote. It is implemented on AVRORA [69], a sensor node emulator, where based on energy measurements from each hardware component, it estimates the overall lifetime of a node. PowerSUNSHINE [70], associated with SUNSHINE [71], is an emulator based on TOSSIM, which also incorporates a hardware simulator. PowerSUNSHINE computes the energy consumption of a sensor node tracking the energy consumed by each hardware component. Cooja [65], the simulator/emulator available in Contiki [17], integrates MSPSim to emulate WSN applications at the instruction level for compatible platforms (i.e., TelosB), and it is able to estimate duty cycles and nodes energy consumption. Simulation-based approaches, despite incorporating accurate hardware models in some cases, are still very sensitive to network dynamics, and thus their usage for evaluating practical WSN data collection applications is limited.

Authors of [72] compute the energy consumed by sensor nodes based on electric current measurements for each hardware component and calculations of the time it has been in operation. These calculations are done as part of the WSN application and report an estimate of the energy consumed. However, the accuracy of these estimations depends on multiple varying factors, and thus, a more detailed instrumentation in the WSN application is required for tracking the energy consumed by hardware components. Further details about the uncertainty in these estimations are discussed in the next section.

Other alternatives for estimating the energy consumption in WSN nodes incorporate additional hardware to provide more accurate readings during a WSN de-

ployment. SPOT [73] presents a scalable power observation tool that attaches an additional board to the motes and allows capturing energy measurements at a node level. Similarly, [74] proposes an integrated testing infrastructure, which incorporates additional hardware for evaluating the power consumption of motes under realistic conditions. iCount [75] provides energy metering by counting cycles of a node switching regulator and it enables current consumption monitoring in real time. By performing energy measurements during the deployment, these alternatives consider the effect of the external environment and network dynamics into the WSN application, and therefore in their final energy estimations; however, using additional hardware also introduces new variables into consideration (i.e., equipment calibration) and it may not always be a feasible option due to power, space, budget, or design restrictions.

6.2 Method

Estimating the energy consumed by WSN nodes, as defined in (6.1), requires accurate measurements of *voltage*, *current* and *time*. In addition, these calculations could be validated by comparing the observed and computed node lifetimes for a given battery capacity, an approach followed in related in works presented earlier. However, these calculations depend on multiple non-controlled variables (i.e., network dynamics, age of the batteries, etc.), which necessarily introduce significant uncertainty into the obtained results. Some sources of uncertainty are as follows:

$$E = V \cdot I \cdot t \tag{6.1}$$

- Mote hardware defines a voltage operating range, in our case within 2.7 V and 3.6 V, which in practice is observed from 2.5 V to 4 V. As the node voltage changes, the current consumed also changes; therefore, assuming a constant voltage of operation during a node lifetime will necessarily affect the accuracy of the results.

- Mote batteries also represent an important source of uncertainty. First, using only two AA batteries underuses node resources because their voltage will be too close to the hardware lower limit of operation. Moreover, when nodes stop working, their batteries have not necessarily consumed all their capacity and the remaining value must be estimated.

- By using three AA batteries per node, the voltage operating range is better used and more of the available battery capacity would be consumed. However, when AA batteries are recharged (in our case NiMH rechargeable batteries), the voltage obtained is approximately 1.5 V; then, three recently charged batteries would exceed the hardware operating range. From our experience, it has been noticed that recharged batteries can be used after a few days, when they self-discharge to a voltage close to 1.3 V. Still, the battery capacity consumed by a node is a critical factor in the accuracy of estimating the node lifetime and small variations in the capacity estimation might produce very different lifetime results.

We consider that the energy consumed by WSN nodes is the sum of the energy consumed by each state of the application for a given time period. Our data collection application defines the following states: sensor sampling, data packet transmission, data packet forwarding, data packet re-transmission, routing packet transmission, routing packet reception, idle listening, and sleeping. It should be noticed that the forwarding state aggregates the energy consumed in both communication actions: packet reception and transmission. For estimating the energy consumption profile, we focus on relative differences between the energy consumed by different components in a sensor node, which correspond to the above defined application states. Therefore, instead of attempting to provide exact values of the energy consumed, we intend to characterize the behavior of a real-world WSN application and its impact on the energy profile and node lifetime.

To this end, our approach to the energy profile estimation is based on two major elements. First, we propose to use the WSN health and instrumentation information obtained from sensor nodes deployed at a real-world testbed. This information incorporates the effect from changes in network dynamics, providing real statistics for each application state. The second element corresponds to the electric current measurements for each state of the application, which can be obtained in a laboratory experiment beforehand.

In our case, the current consumed by MICAz and IRIS motes was measured using an Agilent Technologies DSO7014B oscilloscope and an Extech digital multimeter. These values were obtained for each hardware platform, application version (regular and relay node), and for different voltages within the operating range. Then, the average electric current was computed for each application state.

6.3 Experiments

At the time of the experiment, 50 WSN nodes were available at the ASWP testbed introduced in Chapter 3. These nodes corresponded to MICAz and IRIS motes, each one equipped with an MDA300 data acquisition board. The MDA300 provides embedded temperature and humidity sensors, in addition to ADCs for connecting external sensors, powered through the board's excitation pins. Three types of external sensors are used for monitoring environmental variables: EC-5 soil moisture sensors [76], MPS-1 dielectric water potential sensors [77], and custom made SAP flow sensors [78]. All motes were powered by three NiMH AA rechargeable batteries with a nominal capacity of 2700 mAh. As SAP flow sensors require separate energy sources, they are not considered in this work. The sink node is an IRIS mote with a permanent power supply.

Nodes run a periodic data collection application based on TinyOS 2.1.2, which incorporates the original version of CTP [21] and LPL as described in Chapter 3. Sensor data packets are sampled every 15 minutes and they include all sensor readings,

i.e., temperature, humidity and external sensors. The application was extended to provide node health and instrumentation information for monitoring purposes. This information is collected by introducing additional fields in the sensor data packets and also by generating an additional summary packet every 30 minutes. For LPL, all nodes are configured with a wakeup interval equal to 1 second and default values were used for the remaining parameters.

Two different versions of the application were configured for *relay* nodes and *regular* nodes, respectively. Relay nodes only have embedded sensors and are flexible in their location. We disabled all components controlling the ADCs on the MDA300 in relay nodes for a more energy-efficient operation. Regular nodes do have external sensors attached through the data acquisition board, and thus, their location is fixed.

The application installed at the sink node is configured as the root of the collection tree. It receives packets and transfers them over the serial interface to the WSN gateway. In addition, a special LPL configuration is implemented in the sink node, where it is continuously awake for incoming packets, but still uses the LPL preamble for all its outgoing transmissions.

6.4 Results

Experiments with the oscilloscope allowed us to confirm an important difference between relay and regular nodes when sampling external sensors via the acquisition board to generate a data packet. For regular nodes, it was noticed that the current consumed when sampling these sensors may increase over nine times, as more sensors are connected. This behavior is depicted in Figure 6.1, where approximately at 0.1 s, the node starts sampling external sensors by activating the excitation pins, and the process continues until approximately 0.44 s. Then, embedded sensors are sampled, and the node finishes with a data packet transmission to the sink. Unlike regular nodes, when relay nodes generate a data packet, they only sample on-board sensors

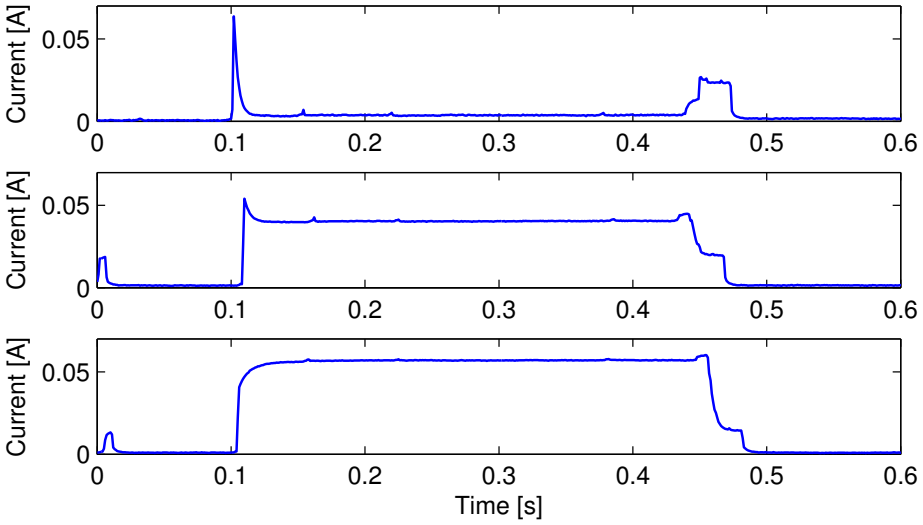


Figure 6.1. Data packet generation for an IRIS node with ADCs enabled on the MDA300 acquisition board: (1) no external sensors (top); (2) two external soil moisture sensors (middle); (3) three external soil moisture sensors (bottom).

and transmit the information with a similar behavior as the top curve in Figure 6.1 between 0.44 s and 0.5 s.

From Figure 6.1, it can also be seen that the energy consumed by regular nodes while sampling external sensors far exceeds the energy consumed for transmitting a packet when the LPL preamble is not needed. Most regular nodes at the ASWP tested require using the LPL preamble because they are located more than one hop away from the sink. In such cases, energy consumed by packet transmissions is also determined by the relative synchronization between sender and receiver; but still, every time a data reading is sampled, a significant energy burden is added to the motes. These observations from IRIS nodes are consistent with the results obtained for MICAz motes.

The average of the electric current measurements obtained for both mote platforms over their voltage operating range are summarized in Table 6.1. These results include the current consumed while sampling the ADCs on a regular node without

Table 6.1.
Electric current measurements

	MICAz		**IRIS**	
	I [mA]	**t [ms]**	**I [mA]**	**t [ms]**
Sample start (0 Ext. Sensors)	56.7	6.5	54.9	6.5
Sample read (0 Ext. Sensors)	6.8	337.0	6.3	341.8
Sample start (3 Ext. Sensors)	67.8	6.5	54.9	6.5
Sample read (3 Ext. Sensors)	91.8	337.0	59.1	341.8
Tx Start	9.2	2.4	6.5	1.5
Tx (no LPL)	22.9	26.5	22.0	25.1
Rx	22.7	24.0	21.1	23.6
Idle listening start	7.4	2.3	7.2	1.2
Idle listening	23.3	3.7	19.1	5.4
Sleep (relay / regular)	0.3 / 1.1	–	0.1 / 1.0	–

sensors and with three attached sensors. Basic transmissions (i.e., without using the LPL preamble) and packet receptions are included, in addition to idle listening. The table also includes the stable sleep current obtained when no activity was detected from the application states. It was noticed that the sleep current in regular nodes is considerably higher compared to relay nodes in both IRIS and MICAz platforms; therefore, even when using low sampling rates, it is expected that regular nodes consume more energy than relay nodes with similar traffic loads. When comparing results obtained between platforms, overall, IRIS motes consume slightly lower currents in shorter periods of time, leading to lower energy consumption. Idle listening on MICAz motes is an exception because despite using a higher current, its duration is shorter, leading to lower energy consumption compared to IRIS motes.

Table 6.2.
Mote parameters and configurations

Node ID	Location	Platform	Batteries	Type	Ext. Sensors
1001	ASWP	MICAz	3	Relay	–
1100	ASWP	IRIS	3	Relay	–
2003	ASWP	IRIS	3	Regular	3 Soil Moisture.
5053	ASWP	MICAz	3	Regular	3 Soil Moisture

6.4.1 Energy Profiles

Nodes with different characteristics were chosen from the ASWP testbed to compute the energy profiles. Their parameters and configurations are summarized in Table 6.2. These nodes represent different locations of the network, each one with specific traffic conditions, as presented on Figure 6.2.

A dataset with packets collected from the tested between November 2013 and February 2014 was selected. Calculations are based on the number of generated packets, received and forwarded packets, re-transmissions, routing packet transmissions, and routing packet receptions obtained from the health and instrumentation information. Then, daily average values were computed and organized with the electric current measurements to determine the energy consumed by each application state. Afterwards, the aggregated active time of the application states was subtracted from the total time to estimate the energy consumed by the motes while sleeping. Since regular nodes have a higher sleep current compared to relay nodes (due to ADCs on their data acquisition boards), the energy consumption increment caused by the difference on the sleep current was accounted as being consumed by the regular nodes data sampling state. This allows us to directly compare sleep states between regular and relay nodes.

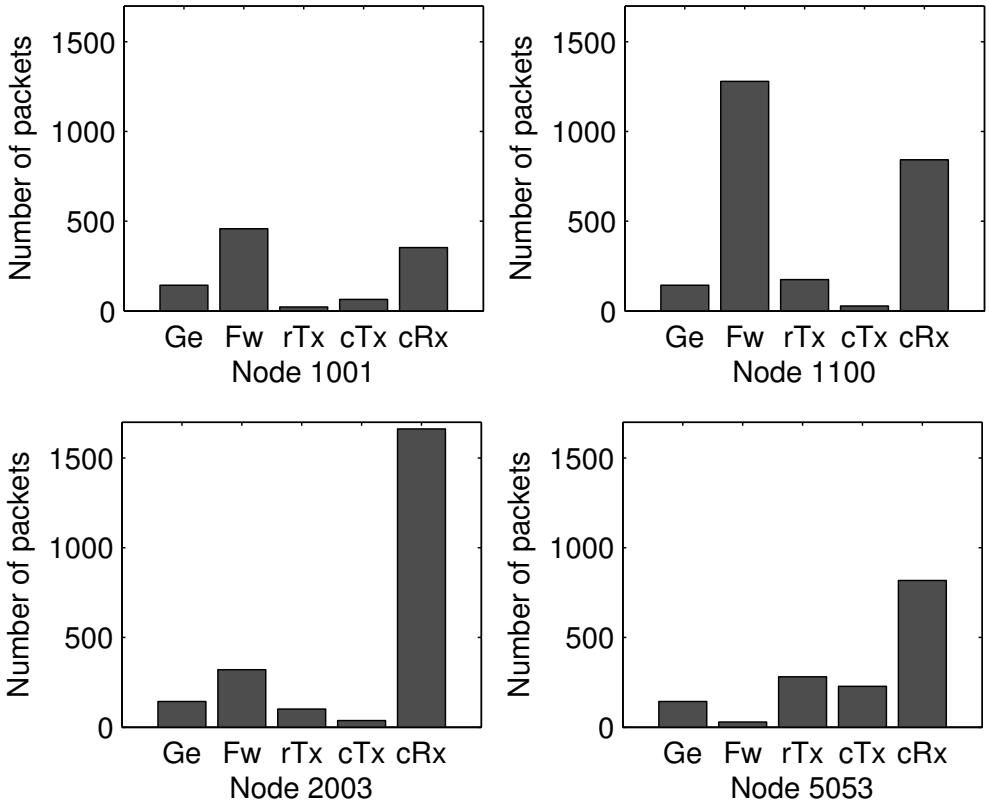
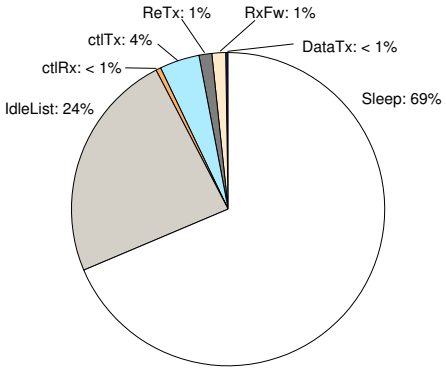
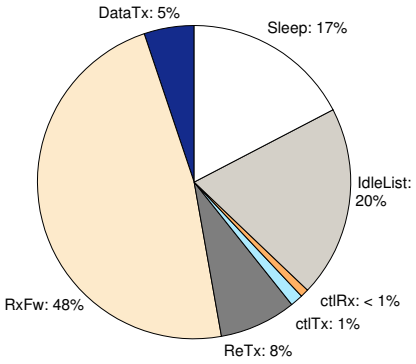


Figure 6.2. Traffic characteristics of selected nodes from the ASWP testbed. Generated data packets (Ge), forwarded/received data packets (Fw), data packet retransmissions (rTx), routing packet transmissions (cTx), and routing packet receptions (cRx). Daily average values are presented.

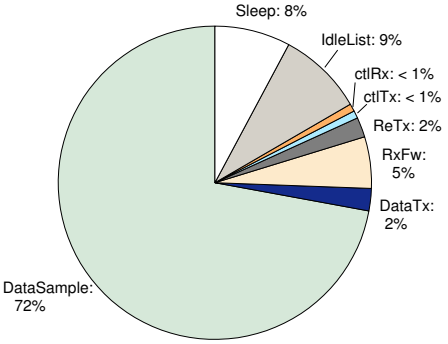
Final results of the energy profiles are presented in Figure 6.3. Relay node 1001 is the only node with a direct communication link to the base station among these four selected nodes and this effect can be seen when comparing its energy profile with that of the other relay, node 1100. While node 1100 forwarded less than three times the traffic of node 1001, its forwarding energy consumption percentage is 48 times that of node 1001, resulting in a difference of 22,979 mAs/day. This observation clearly shows that packets transmitted from nodes that can directly reach the base station have a much lower impact on their energy consumption. On the other hand, relay nodes that require using LPL preambles, i.e., node 1100, are more sensitive



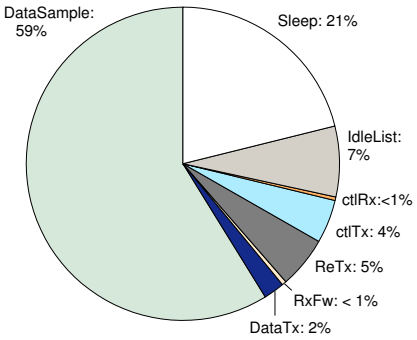
(a) Node 1001. Duty cycle: 0.7% (37481 mAs/day)



(b) Node 1100. Duty cycle: 2.2% (48654 mAs/day)



(c) Node 2003. Duty cycle: 1.3% (109110 mAs/day)



(d) Node 5053. Duty cycle: 1.4% (120978 mAs/day)

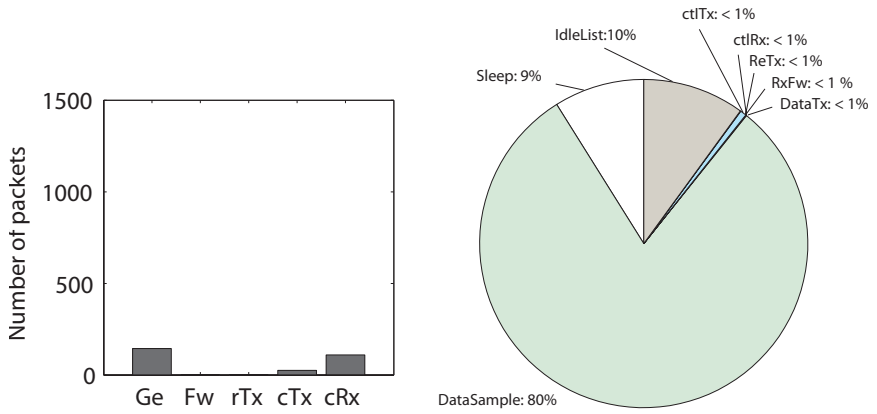
Figure 6.3. Energy profiles and duty cycles of selected nodes at ASWP. Daily average values are presented for the following application states: data sampling, data transmissions (DataTx), data receiving+forwarding (RxFw), data retransmission (ReTx), routing transmissions (ctlTx), routing receptions (ctlRx), idle listening, and sleeping.

to network dynamics, where forwarded packets, retransmissions, and routing packets could account for most of the energy consumed.

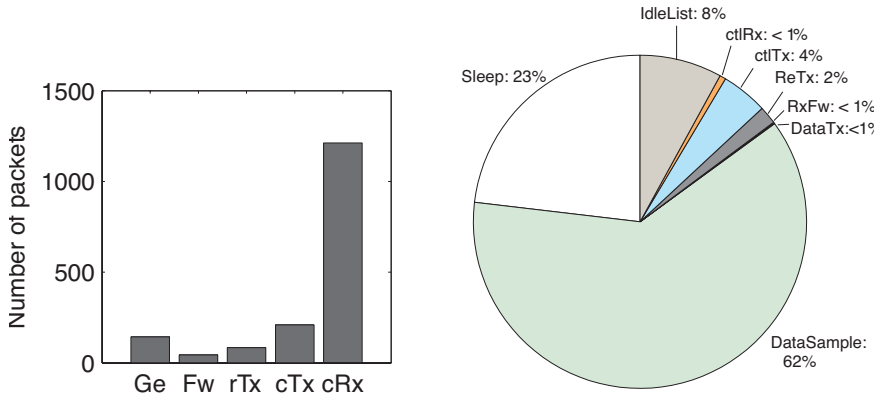
For regular nodes 2003 and 5053, it can be seen that most of the energy consumed is a result of sampling the external sensors. However, the main cause for these high percentages is the increment in the sleep current for this configuration, which accounts for over 95% of the energy consumed by the data sampling task in the energy profiles. If regular nodes could use a more efficient MDA300 driver/hardware, which could maintain a low sleep current when the ADCs are enabled, the energy profiles of regular nodes would be more consistent with those obtained from relay nodes. Furthermore, the effect of sampling the external sensors presented in Figure 6.1 would also have a higher impact on the nodes energy consumption and lifetime.

We note that receptions, in general, have a low effect on the energy profile. Even after receiving high routing traffic, as in node 2003, or when considering the energy consumed to receive each forwarded packet; the overall energy consumed is not significant compared to other application states. This is a direct result of the asynchronous LPL configuration, which requires a higher effort from sending nodes. Another consequence from this is the effect of link quality into the energy profile. In Figure 6.2, node 5053 shows the highest number of retransmissions, relative to the number of generated and forwarded packets. This indicates lower link quality, which at the same time is making data transmissions much more expensive. For this node, for example, if an optimal acquisition board and driver were used (i.e., with a negligible effect on the sleep current), the total data packet transmissions (including retransmissions) could add up to 25% of the total energy consumption.

The approximate duty cycle of the nodes, computed based on the active time of the transceiver, is within 0.7% and 2.2%, as shown in Figure 6.3. It can be seen that higher traffic conditions can greatly increase the duty cycle of a relay node. Results for regular nodes evidence that duty cycles alone are not enough for understanding the energy consumption in WSN nodes. As shown in the figure, the effect of external



(a) Node 106. IRIS regular node, 0.7% duty cycle, 96375 mAs/day



(b) Node 107. MICAz regular node, 1.0% duty cycle 110922 mAs/day

Figure 6.4. Results obtained for nodes in a laboratory experiment. Daily average values are presented. Drivers are enabled in these nodes, but the external sensors are not attached. They use two AA batteries of the same reference as nodes at ASWP.

sensors greatly impacts the energy profile, but the duty cycle may still be lower compared to relay nodes.

6.4.2 Node Lifetime

The lifetime of WSN nodes can be calculated based on their energy consumption and their available battery capacity. However, there are many factors affecting the performance of the batteries (i.e., number of charging cycles, age, and temperature)

Table 6.3.
Expected node lifetime from laboratory experiments

Consumed Battery Capacity	**Node 106**	**Node 107**
40%	38 days	33 days
50%	48 days	42 days
60%	58 days	50 days
70%	67 days	59 days
80%	77 days	67 days
90%	87 days	75 days
100%	97 days	84 days

that these estimations would be unreliable. A controlled laboratory experiment was performed to estimate the lifetime of regular nodes deployed using a similar configuration to that of nodes at the ASWP testbed. In the experiment nodes are powered using two fully charged batteries and Figure 6.4 presents their energy profiles, duty cycles, and consumed battery capacity. The observed lifetimes for nodes 106 and 107 in this test were 49 days and 39 days, respectively. Then, we assume that nodes are able to consume different percentages of the battery capacity and compute the expected node lifetimes based on their energy profiles, as shown in Table 6.3. It can be seen that a difference of 10% in the consumed capacity increases the expected node lifetime by 8 days or more, affecting the accuracy of the lifetime estimations. In this laboratory experiment, nodes consumed around 50% of their battery capacity.

The expected lifetimes from nodes at the ASWP testbed are shown in Table 6.4 based on the energy profiles from Figure 6.3 and for different consumed battery capacities. In the testbed, regular nodes 2003 and 5053 depleted their batteries after 75 and 63 days, respectively. On the other hand, the batteries of relay nodes 1001 and 1100 were replaced after 143 days, but before they were depleted. Based on the observed lifetimes, regular nodes in the testbed consumed over 80% of their capacity,

Table 6.4.
Expected lifetime from nodes at the ASWP testbed

Consumed Battery Capacity	**Node 1001**	**Node 1100**	**Node 2003**	**Node 5053**
40%	99 days	76 days	34 days	31 days
50%	124 days	96 days	42 days	38 days
60%	149 days	115 days	51 days	46 days
70%	174 days	134 days	60 days	54 days
80%	199 days	153 days	68 days	62 days
90%	224 days	173 days	77 days	69 days
100%	249 days	192 days	86 days	77 days

while relay nodes 1001 and 1100 had consumed under 60% and 80% of their capacity, respectively. The differences in the consumed capacity compared to the laboratory tests are caused by the number of batteries used in each scenario, since nodes using two AA batteries are operating closer to the hardware lower limit of operation, as explained earlier in the chapter. In addition, the batteries of the laboratory tests had been used more frequently and had more charging cycles, reducing their expected capacity. Finally, for nodes deployed at the testbed, it is also important to consider that the data was collected during the winter season and nodes were exposed to freezing temperatures, which also reduce the expected capacity of the batteries.

CHAPTER 7. INTEGRATED NETWORK AND DATA MANAGEMENT SYSTEM FOR HETEROGENEOUS WSNS

WSN management becomes increasingly important to monitor and ensure that deployed motes operate correctly and healthily along time. The resource constraints of WSNs have introduced and involved different hardware and software technologies of sensor networking, being designed for very specific purposes. As a result, users in multiple applications are directly facing the complexity of interacting with diverse technologies from different manufacturers and specific requirements [5]. Indeed, the emergence of multi-platforms and their different management systems for WSNs that co-exist in different deployment sites has made effective network and data management become even more challenging. We refer to this type of networks as *Heterogeneous WSNs*.

Moreover, in practice, WSN management operations such as monitoring, configuration and maintenance should not affect the main application running on the network. This situation applies especially for long term solutions, in which the cost and complexity of network management operations may force an application solution to be unfeasible or cost-ineffective [79].

In this chapter, we present the design and implementation of the infrastructure to support network and data management for heterogeneous WSNs. To this end, we present a web-based *Integrated Network and Data Management System for Heterogeneous WSNs (INDAMS)*. Our system is a framework for heterogeneous WSN management centered on three fundamental design criteria: (1) systematically supporting heterogeneous WSNs with a unified system; (2) clearly separating WSN management functions from WSN applications; and (3) easily accessible web-based user interface for management functionalities. In this way, our system is able to effectively support a variety of applications deployed in multiple WSNs from different administrations

that could also involve diverse WSN platforms and technologies. Furthermore, users are able to access the management system independently, retrieving any information and monitoring any WSN(s) operations with a unified set of tools, without dealing with the specific details and complexity of underlying WSN gateway commands and configurations. Thus, our system provides a unified management framework for users with different underlying WSN platforms and technologies. In addition, newly deployed WSN(s) at new site(s) can easily join this unified management system with minimal effort. Figure 7.1 illustrates the general architecture of the system, where multiple WSNs are connected to a management server which multiple users can remotely access for their WSNs.

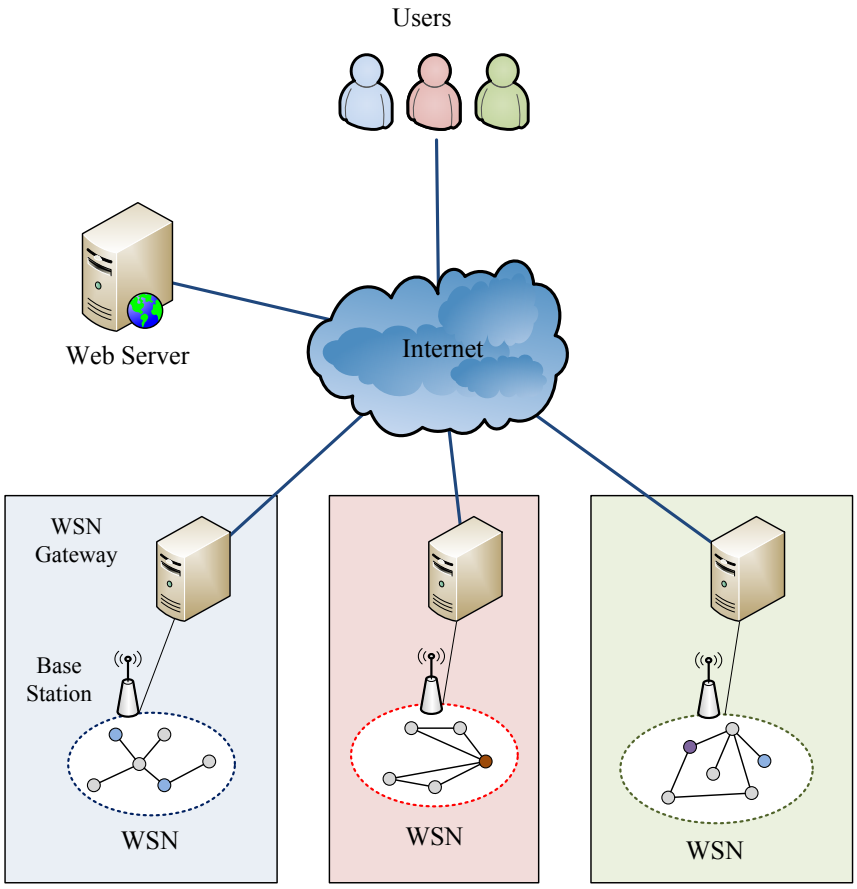


Figure 7.1. An illustration of the management system general architecture.

7.1 Related Works

Multiple works have been reported for network management in WSNs [80]; however, many of them are limited to specific networks or applications. MOTE-VIEW [42] is a sensor network monitoring and management tool included in MEMSIC's MoteWorks software platform. MOTE-VIEW provides three main functionalities related to data persistence, network-health monitoring, and data visualization. In addition, it offers a graphical interface for functions provided by other tools also included in MoteWorks, i.e., XServe and XServeTerm. However, MOTE-VIEW's strengths come from its tightly coupling with MoteWorks proprietary platform, which limits the possibility for extending and integrating with other WSN platforms and tools.

Various related works in WSN management are focused on protocol design or provide extensions to specific network management protocols. Authors of SNMS [81] propose a sensor network management system based on their previous experience with WSN deployments. SNMS is intended to run in WSN nodes alongside with the main application. This system allows user-defined queries for specific data, e.g., battery voltage, and it also offers a logging function for generated events. Moreover, SNMS targets an application-independent operation and thus implements its own lightweight network layer. BOSS [82] presents a system architecture for supporting the Universal Plug and Play (UPnP) protocol in WSNs. In their work, the base station implements an UPnP agent and bridges the WSN with an UPnP network. MannaNMP [83] defines the WSN management protocol based on MANNA [84], a policy-based management system architecture. This protocol defines information exchange among management entities, following a cluster-based approach for organizing the WSN nodes. Additional related management protocols are proposed in [85] and [86]. Compared to these works, INDAMS follows a different approach intending to support multiple and heterogeneous networks in a unified system, and therefore, it does not constraint WSN applications with specific protocol or application requirements.

Octopus [87] is a sensor network monitoring, visualization, and control tool characterized for being open source and protocol independent. Octopus is a standalone management application with different configuration options for sensor motes and it focuses on a single WSN management environment. Protocol independence is a common characteristic between INDAMS and Octopus. However, INDAMS extends these functionalities by supporting multiple and heterogeneous WSNs in a unified management framework. Furthermore, INDAMS not only addresses network management functions, i.e., monitoring, visualization and control, but it also incorporates data management functions.

Other related works on management systems for WSN include those reported in [88] and [89]. These two systems offer generic application environments developed in .NET and Java, respectively. They aim to provide a web-based system for WSN management. Work presented in [88] focuses more on providing a flexible and extensible web interface, but is limited to single-network management scenarios. The work of [89], called jWebDust, focuses more on the network system architecture with the introduction of the concept of virtual sensor network, abstracting multiple networks into a single virtual WSN. The main difference between INDAMS and jWebDust is that our system supports multiple WSN administrations, and recognizes the differences between multiple heterogeneous WSNs. In our management system, multiple WSNs are not viewed as virtually the same; they are differentiated according to their specific characteristics but inside a unified management framework with the same sets of management tools.

7.2 Management System Architecture

In order to address WSN management heterogeneity introduced by multiple platforms, diverse applications, and technologies, we propose a layered system architecture as follows.

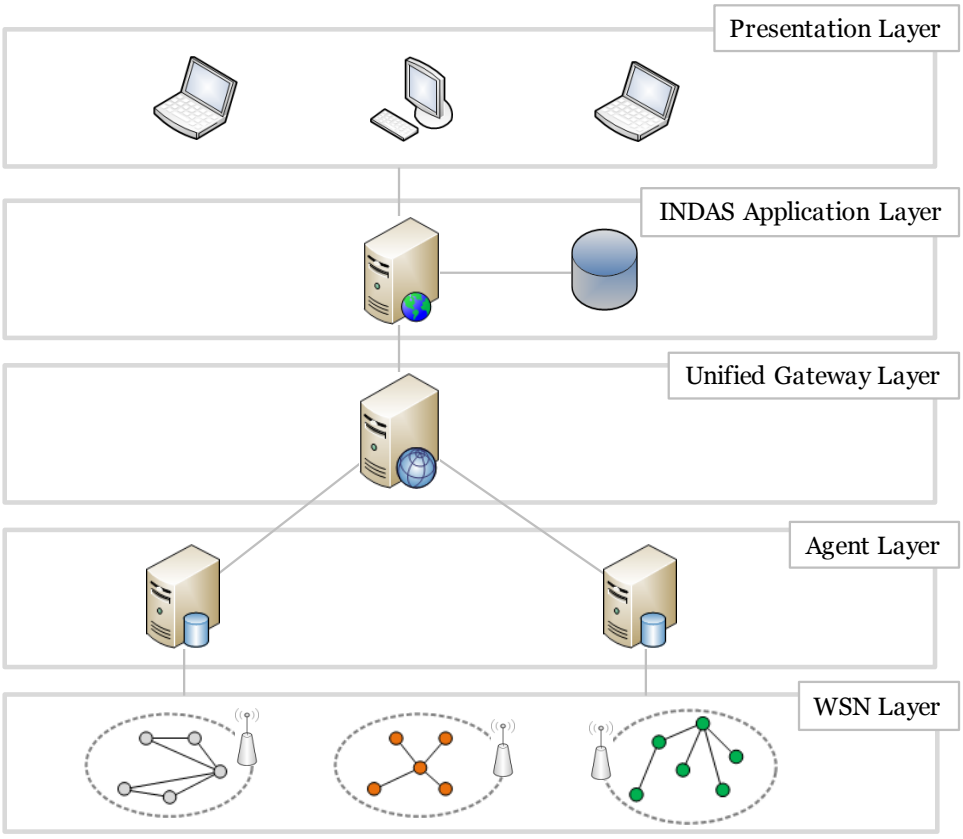


Figure 7.2. INDAMS layered architecture.

7.2.1 Layered Architecture

The architecture, shown in Figure 7.2, allows a logical separation between different functions, hiding their complexity to the upper layers.

1. *Presentation Layer:* This layer is in charge of user-system interaction. It is implemented in a web interface that captures the information to be processed by other layers and displays the processing results.

2. *INDAMS Application Layer:* It processes the information received from the presentation layer and implements the interface mechanisms to communicate with lower layers.

3. *Unified Gateway Layer (UGL):* It corresponds to the most important layer in the architecture, as it specifies a unified communication interface with all individual WSNs, and thus forms an abstraction level that hides management complexity from all heterogeneous WSNs. This enables the definition of unified management functions at this layer regardless of any heterogeneity existing in the underlying individual WSN management commands.

4. *Agent Layer:* It is introduced as the middleware that communicates the Unified Gateway Layer with individual WSN Gateways (WSN Layer). An agent works directly with a local WSN gateway, allowing our management system to handle multiple and heterogeneous WSNs in an abstract way. A key part of our system is the communication between the Unified Gateway Layer and the Agent Layer, which is defined later in this chapter. The agent layer is in charge of implementing technology-specific functions associated with individual WSN platforms to communicate with different WSN gateways.

5. *WSN Layer:* This bottom layer corresponds to a concrete WSN deployment. The network is controlled by one or multiple WSN gateways that provide interfaces to and from motes for control commands, operational states, and data communication.

7.2.2 Components View

The components view of INDAMS architecture is presented in Figure 7.3. This figure indicates the main components present in INDAMS server and agent terminals. INDAMS server receives user requests through the web interface and they are processed by Controller components. Controller components receive requests from multiple clients, and send them with the right parameters to the Server Unified Gateway (UG). Similarly, the Data Handler receives data from the Server UG and distributes them to the corresponding destination(s).

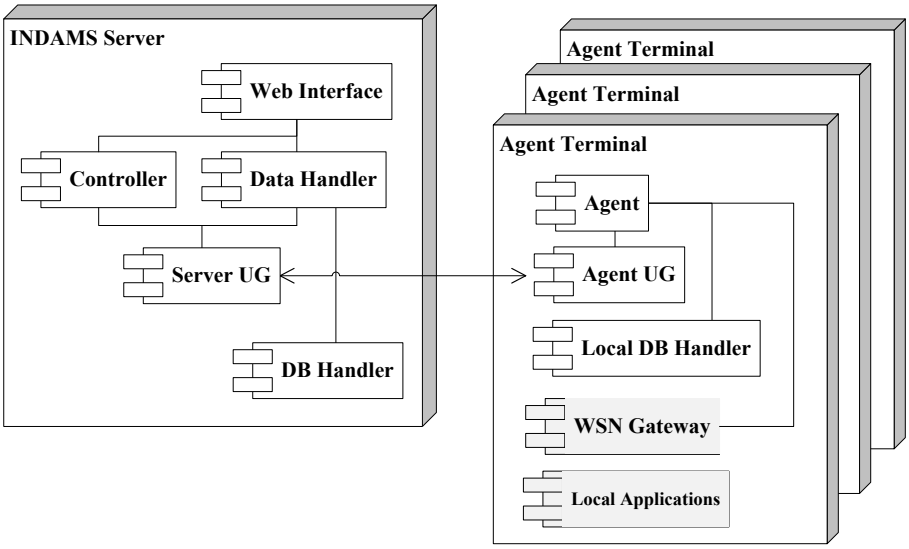


Figure 7.3. INDAMS components view.

Agent terminals also execute the WSN gateway and local applications, i.e., DBMS, remote connection applications, etc. On the agent side, the Agent Unified Gateway (UG) controls all communication with INDAMS server and the Agent component controls the interaction with the WSN gateway and local database. Controller and Data Handler components serve clients, whereas the Server UG serves agents, as illustrated in Figure 7.4.

In the proposed layered and flexible architecture, the concept of clients is not limited to web users. A client can be defined, in this context, as any user or application that is able to perform requests and wants to receive data from WSNs. Hence, a client does not only refer to a web application or a user, but it may also refer to a database configured for a specific WSN, an external application, or a logging function, among other alternatives. That is, clients represent various subscribers for WSN data. A publisher/subscriber approach is adopted in our design and implementation.

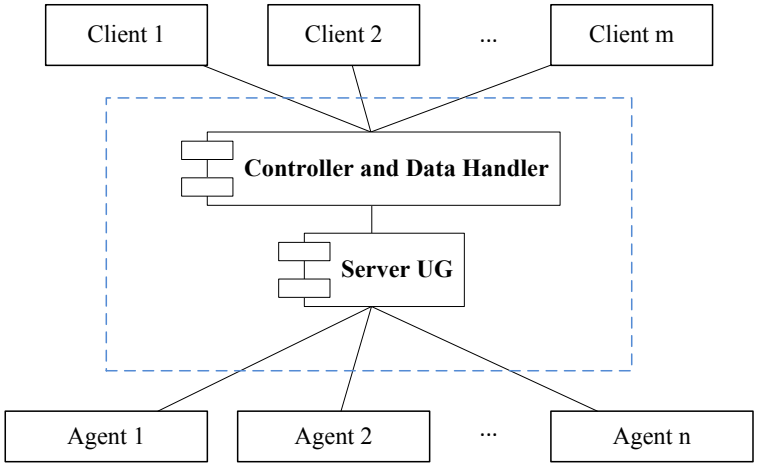


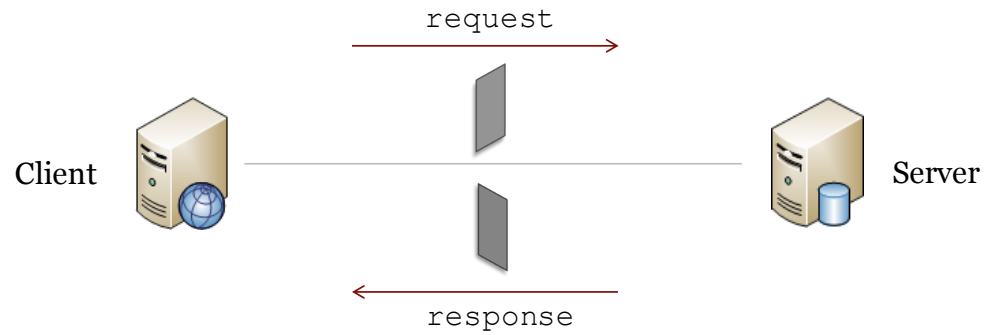
Figure 7.4. Control/Data Handler and server Unified Gateway (UG).

7.2.3 Agent Functions

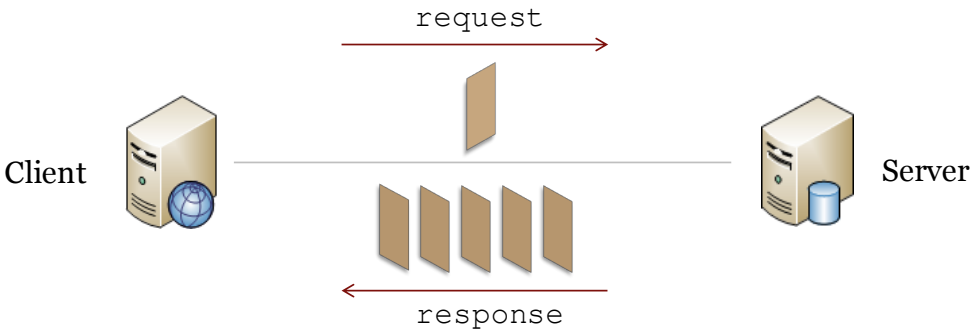
For supporting a broad range of WSN applications, we classify management system functionalities at the UGL into two categories: *request/response functions*, and *continuous data functions*. Request/response functions are mainly used as control commands, either WSN commands (i.e., set node 10 to sleep) or agent commands to control continuous data functions (i.e., start/stop data collection). On the other hand, continuous data functions represent any action which would result in a continuous data stream, e.g., data collection and object tracking. These categories apply for both WSN management and application functions, since the UGL is not aware of this differentiation. A graphical representation of function categories at the UGL is presented in Figure 7.5 for a generic client/server scenario.

7.2.4 User Access Control

INDAMS is designed to support different users, from multiple organizations, trying to access their WSN information. As more users register and use the system, it is of



(a) INDAMS request/response function.



(b) INDAMS continuous data function.

Figure 7.5. Graphical representation of function categories at the UGL for a generic client/server scenario.

critical importance to define a mechanism for controlling their activities; otherwise, important experiments or continuous operations could be altered or interrupted by users exploring the system or by unintended actions.

Access control in INDAMS is based on *WSN agents*, *agent functions*, *users* and *user roles*. WSN agents define a group of provided functions and each function is associated to a user role. Additionally, user roles are not global to the system, since a determined user could be privileged for one WSN, but a basic user for all other WSNs. Therefore, user roles are defined for each WSN agent. For example, only WSN administrators should be allowed to start and stop the WSN gateway in a par-

ticular agent, but any user in the system could be allowed to monitor the WSN. In this case, the agent data collection function would be associated to the administrator role and the data monitoring function would be associated to the general user role in this agent. After this, users are assigned their corresponding role in each WSN agent. A special role was defined for system administrators, which do not have any restriction and also have access to system level functions, i.e., starting/stopping the system. Figure 7.6 shows the data model used for user access control in INDAMS.

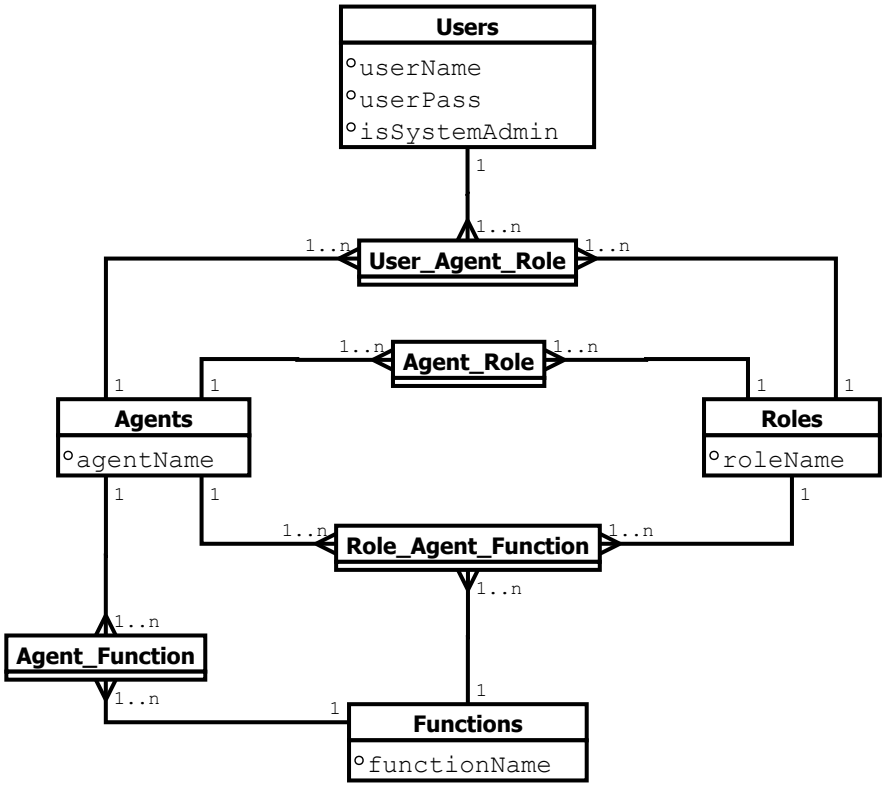


Figure 7.6. Data model for user access control in INDAMS.

7.3 Agent-Server Communication

Communication between server and agents is a key part of INDAMS design and implementation. The initial version of the system defined an Agent-Server Protocol,

which was later replaced by a web-service approach. This section describes both implementations.

7.3.1 Agent-Server Protocol

INDAMS initial agent-server protocol is defined as an application-layer protocol, carried by TCP for reliable transmissions. The main elements of this protocol are as follows.

Registration

The process conducted by the server whenever it receives a new agent connection request is defined as *registration*. During this process, the server and new agents need to exchange all necessary information.

Agent metadata are defined and stored on their side to be easily accessed by the WSN administrator. This information is exchanged during the registration and it describes the WSN controlled by the agent in terms of network size, location, and technology. Network size defines the number of gateways and number of motes; gateway and mote locations are given by longitude and latitude coordinates, and technology is described by the gateway platform (e.g., MoteWorks), mote types (e.g., Mica, MICAz, TelosB), and data acquisition boards (e.g., MTS400, MDA300). In addition to description information, the metadata also include communication parameters like port numbers, server IP address, and any technology-specific parameters required by the WSN gateway platform.

Agent and Server States

The agent-server protocol is designed as a stateful protocol, where agent and server implement a sequence of state transitions and at any given time point both of them know the state of their counterpart. For each UGL function category (i.e., request/response functions and continuous data functions), the agent-server protocol

defines a specific connection: *one control connection* and *one data connection*. State transitions implemented at each agent/sever side control the two connections and information transmissions. This design allows us to set up a general structure of the protocol, capable of supporting a broad range of applications, including management and other application functions.

The overall process starts with the agent initialization, followed by its registration in the system. Once an agent is accepted, it starts waiting for requests generated from any function in INDAMS and forwarded through the protocol server. A request could be either from request/response functions or from continuous data functions. Each request/response function is implemented in a separate specific state, where the request is processed and the corresponding communication between the agent and its WSN gateway takes place. Then, the agent should receive a response from the WSN gateway, and it translates and forwards the response back to the server in a proper format. After this transmission finishes, the agent goes back waiting for new requests.

The implementation of continuous data functions is different, since there are several responses associated to the same function. An example of these functions could be a data collection function. It may accept parameters such as start, stop, update, etc.; then, each parameter may be associated to a request forwarded by the server. The continuous characteristic of this type of functions requires the implementation of concurrent operations in both sides of the protocol.

Figure 7.7 presents a simplified version of the agent state transition diagram. States representing a data collection function, which requires the continuous transmission of data, are highlighted in the figure. These states are associated to the data connection between agent and server, and for this reason they can run concurrently with the other states associated to the control connection.

Regarding to the server side of the protocol, the server implements the corresponding states for each registered agent. In general, the server performs two main tasks. First, it is always waiting for agent registrations; then, when a registration request is received, the server processes it and goes back waiting. Figure 7.8 shows a simplified

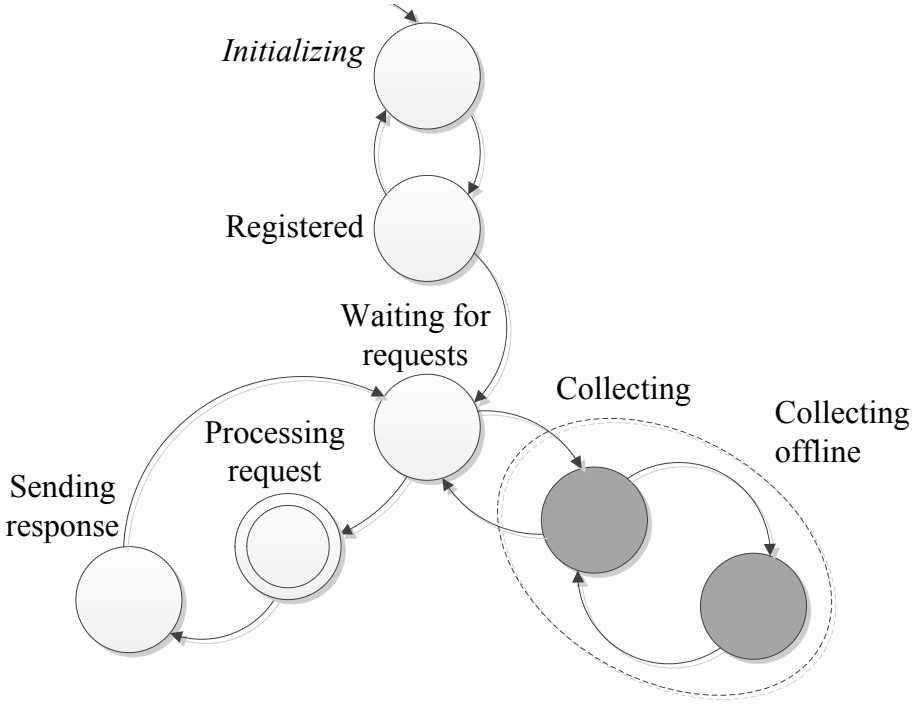


Figure 7.7. Simplified state transition diagram of an agent.

version of the server state transition diagram, corresponding to the version of agent state transition diagram shown in Figure 7.7.

7.3.2 Unified Gateway (UG) Web Service

After using the agent-server protocol for some time, it was noticed that Internet connections to WSN deployment sites are not very stable and trying to maintain permanent connections introduced unnecessary complexity and overhead for the system when attempting to reestablish socket connections. Furthermore, after modifying the agent-server protocol for opening and closing socket connections as needed, agent states were simplified and a stateful representation was no longer needed. In consequence, the protocol implementation was replaced by a web-service alternative.

In this approach, agents support both UGL function categories. An agent defines and control its own state based on continuous data functions (e.g., data collection)

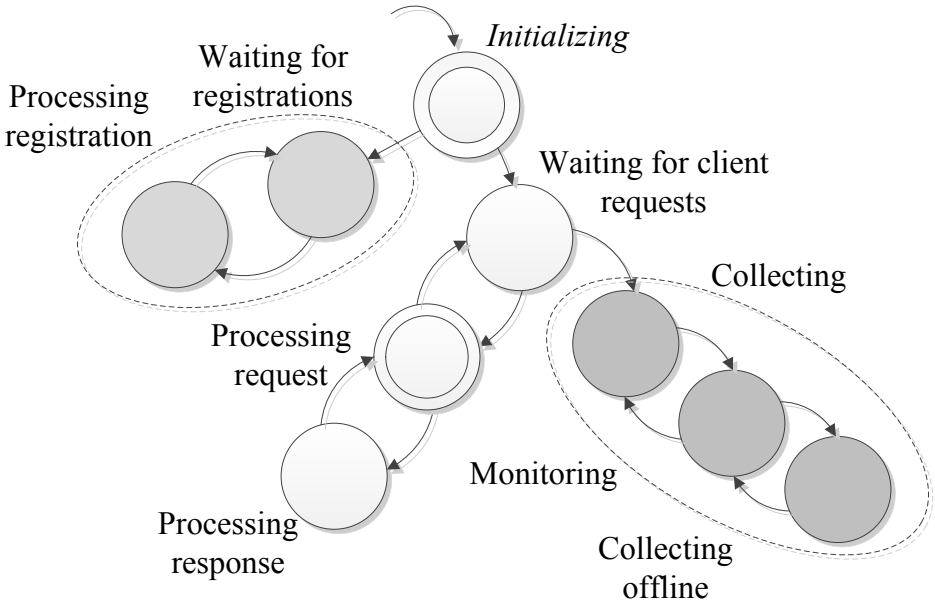


Figure 7.8. Simplified state transition diagram of the server.

and it implements a remote interface as a web service that defines request/response functions. Continuous data functions are controlled through the web service, and they result in concurrent execution threads connecting to INDAMS server. Figure 7.9 show the implementation example of this approach for a data collection function. INDAMS server uses the web service provided by an agent to control the data collection function, which runs concurrently in the *MonitoringThread*. This thread sends the continuous data stream to the *MonitoringController* on the server side. Likewise, agents can implement other continuous data functions, i.e., a heartbeat function.

For simplifying the agent metadata remaining on the agent side, agent description information was moved to the server side. This change allows end-users to access and modify it through INDAMS web interface and only communication parameters are kept for agent configuration. These changes also allow simplifying agents registration

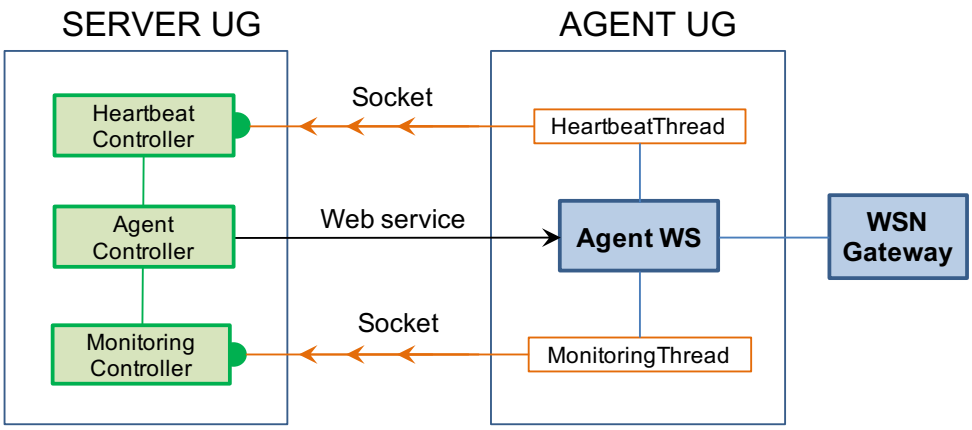


Figure 7.9. Implementation example of the agent-server communication via web services.

to a heartbeat function, which periodically indicates that the agent is running for logging purposes.

7.4 Data Monitoring

Continuous data functions are defined to send a continuous stream of information from the WSN to INDAMS server. These functions are of high interest from multiple users because they enable different options for monitoring in near real time a WSN deployment. Still, each user/client request for these functions should not result in a direct request to the WSN gateway, considering that more than one user may be trying to access the same function at the same time. Figure 7.10 represents this scenario, where multiple clients are trying to monitor the information from the same WSN. Then, two functions are identified: *data collection* and *data monitoring*. As mentioned earlier, data collection is a continuous data function and it would be executing at most once for each WSN. On the other hand, data monitoring is an application-layer function, which does not need to go through the UGL. In this way, every client that wants to access the continuous stream from the data collection only

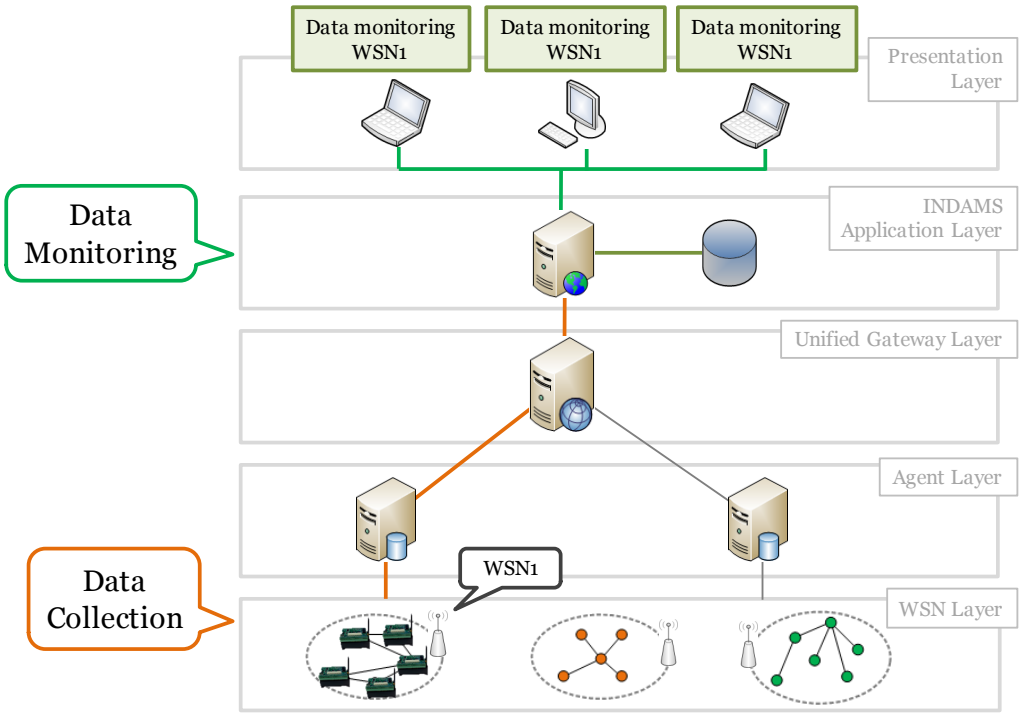


Figure 7.10. Multiple clients trying to monitor the same WSN. The data collection function (orange) goes through the UGL and it is executed only once for each WSN. The data monitoring function does not go through the UGL and each client receives the collected data.

makes a request to the INDAMS application layer and it will start receiving the WSN data.

Client behavior in the data monitoring function describes a publisher/subscriber pattern where each client subscribes to receive the WSN data being published by the data collection function. The data handler component introduced earlier in this chapter implements this approach. An example of the process performed by the data handler is presented in Figure 7.11, where multiple clients (i.e., end users and applications) subscribe to receive the data collected from two WSNs. As seen in the figure, clients are flexible to subscribe to one or multiple WSNs, receiving the data accordingly.

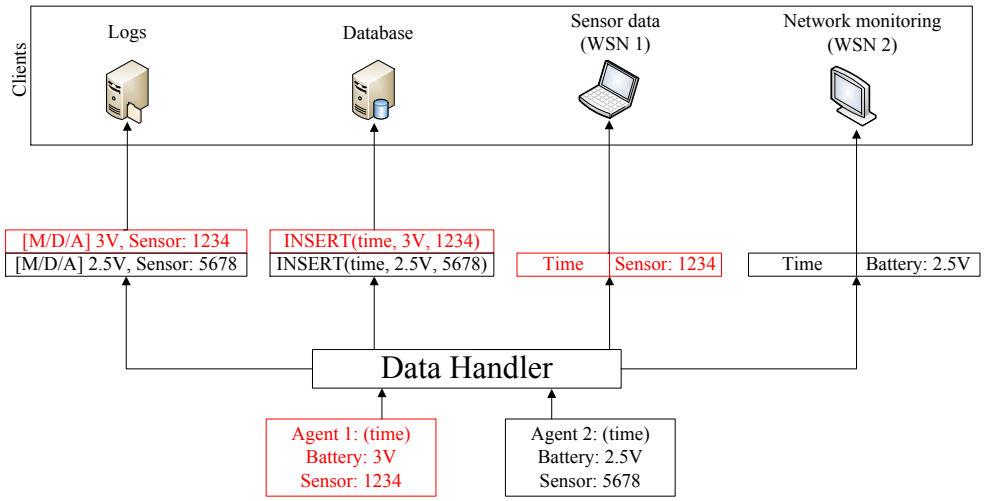


Figure 7.11. Operations of the data handler.

7.5 System Implementation

The system implementation is focused on data collection applications where multiple nodes are deployed in outdoors locations, sensing different variables, and sending their measurements to a sink node (i.e., base station). The sink receives the packets from the entire network, and forwards them to the WSN gateway through a serial interface. The WSN gateway processes the data packets and sends them to INDAMS and local applications if needed (e.g., local database).

INDAMS is developed in Java and integrates other technologies depending on their architectural layer, i.e., jQuery and Google APIs which are used at the presentation and application layers.

7.5.1 Agent for XServe

INDAMS has been designed to support the ASWP testbed presented in Chapter 3, which was initially deployed using MoteWorks platform (XMesh and XServe 2.0) [90]. The agent for XServe WSN gateway was implemented in Java as a standalone application. This agent implements one side of the UGL, the communication interface

to the management system. We studied XServe's functionalities [91] and classified them according to the functions supported by the system. Then, XServe's agent in INDAMS supports data collection, data monitoring, and configuration functions.

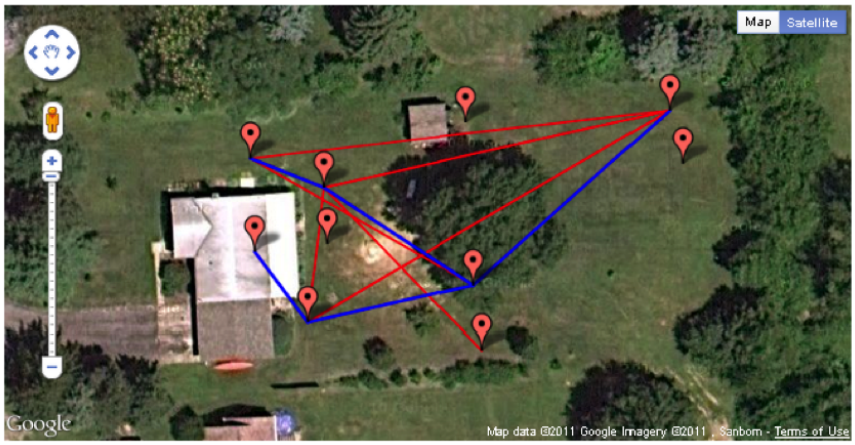
The XServe application runs in a Linux environment and offers a set of parameters to activate different functionalities. We assigned a technology-specific section of the agent metadata to store parameters and values required to start the gateway process. XServe also provides an interface to allow external applications to send configuration commands to the WSN or to the gateway itself. These commands are called XCommands and the application that provides an interface to execute them is called XServeTerm [91]. The definition of XCommands required by the agent is included in the agent metadata to complete the mapping of function requests. In this way, each request received at the agent side can be mapped to a combination of parameters for XServe and XServeTerm, with XCommands and values. Therefore, when the agent receives a request, it checks the metadata for the appropriate mapping and syntax, and communicates with XServe and XServeTerm applications via Java inter-process communication mechanisms.

7.5.2 Agent for TinyOS

A second agent was implementing for TinyOS applications. The WSN gateway presented in [92] provides a flexible way for receiving, parsing, and persisting the received data in TinyOS applications, independently of their protocols and algorithms. Then, the TinyOS agent uses the XML-based communication interface provided by the WSN gateway, integrating it with INDAMS.

Similar to the agent for the XServe gateway, the TinyOS agent receives requests from INDAMS and these are translated to technology specific commands based on the agent metadata. In this case, the WSN gateway is also available in Java and the process communication is done using sockets and an XML-based interface. Currently, the data is persisted directly by the WSN gateway, although the TinyOS agent can

also provide this functionality. This agent supports data collection and monitoring functions and it may also be extended to support additional functionalities (e.g., node configuration and downstream communication) depending on the WSN gateway capabilities.



Monitoring Data : Clear Text

Time Stamp	Packet Type	Node ID	Data	
2011-05-17 16:36:04.534	11	5000	**adc4** 1268.310547 **adc6** 1281.127930 **adc1** 626.831055	**group** 130 **humid** 60.371964 **humtemp** 16.440001

Figure 7.12. An illustration of INDAMS for WSN topology monitoring in a residential backyard.

7.6 Deployment and Web Interface

INDAMS has been deployed for multiple WSN testbeds and laboratory experiments. The first prototype experiment was carried using a small outdoor WSN deployment. This experiment used 10 WSN nodes and one sink located 6 m to 60 m away from one another in a residential backyard in Western Pennsylvania (40.5436 N, 80.0638 W). We tested the agent-server protocol with data collection and monitoring

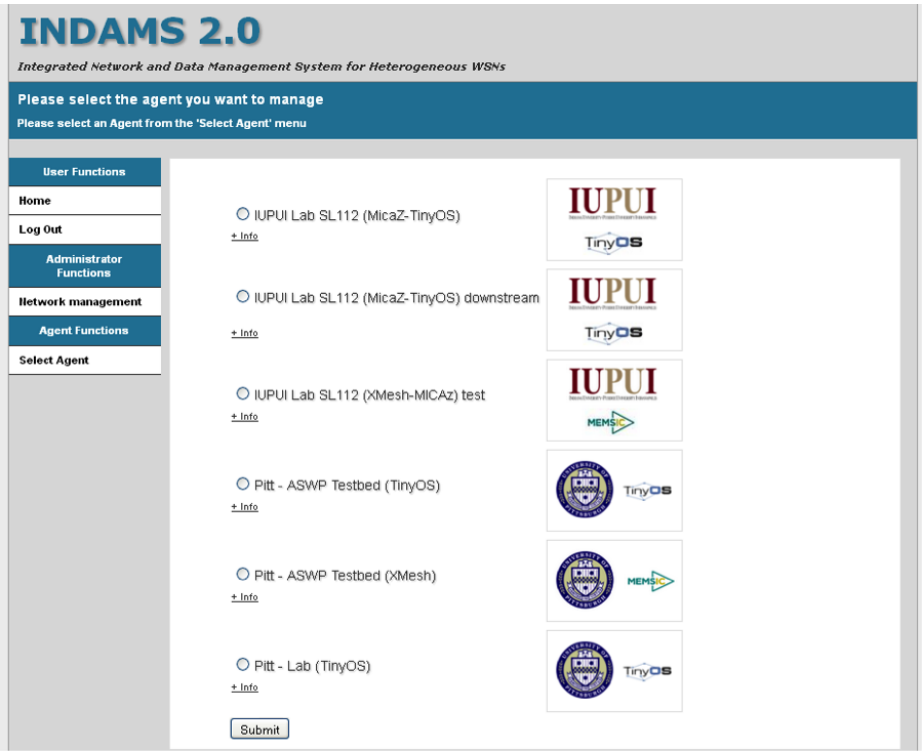


Figure 7.13. INDAMS web interface for agent selection.

functions. This experience allowed us to evaluate the system stability and to identify useful features and potential improvements. For this initial version, a feature called topology monitoring was included in the monitoring function. With this option, users are able to see geographical locations of WSN motes, mote neighbors, and routes used for sending data packets to the base station. The map showing the information is continuously updated as the information is being received at the server side providing a useful tool to have a fast and updated view of the network state. Figure 7.12 shows a screenshot of INDAMS web interface for topology monitoring, in which blue links represent motes selected to forward packets, and red links represent the neighbors for each mote as identified by the routing protocol. The base station and the gateway are located inside the house.

Following this prototype experiment, a stable version of INDAMS was deployed with the ASWP testbed, supporting the subsequent upgrade of the WSN application

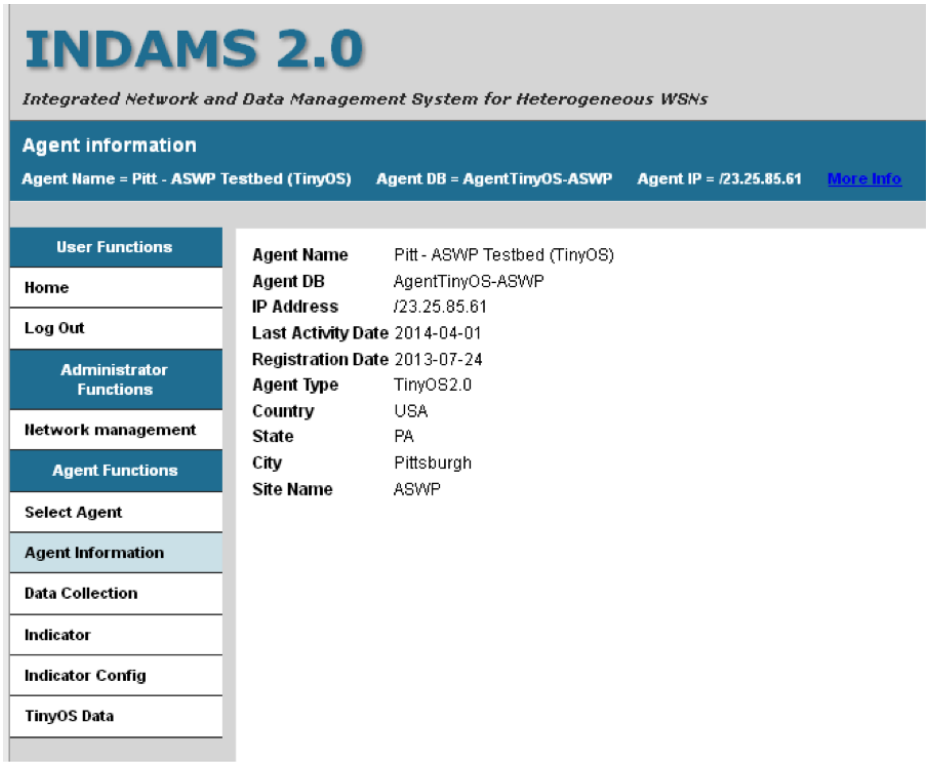


Figure 7.14. An example of the agent functions available for the ASWP testbed.

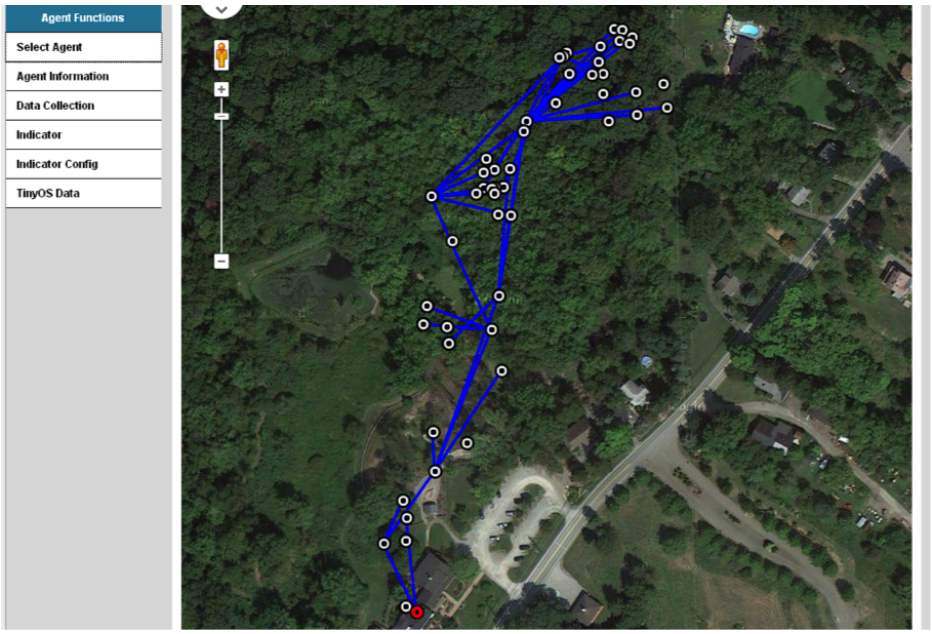


Figure 7.15. Topology monitoring for the ASWP testbed.

based on TinyOS. The web interface in this stable version is shown in Figure 7.13. After users log in, the web interface displays the list of agents available for each user. When an agent is selected, the list of agent functions is updated based on the user privileges; afterwards, any of the available options can be accessed from the menu. Figure 7.14 shows an example of the agent functions available for the ASWP testbed.

Data collection and data monitoring functions have shown to be very valuable for the ASWP testbed. At deployment time, the map indicates if a WSN node starts working correctly and the parent node chosen for packet forwarding, as illustrated in Figure 7.15. This function also facilitates the identification of bottlenecks and highly used nodes in the WSN. Similarly, voltage values, sensor readings (i.e., temperature and humidity), and health statistics can be easily identified from the continuous data stream, as seen in Figure 7.16, allowing for a fast an efficient diagnostic tool that can also be accessed from a mobile device during on-site deployment and maintenance visits.

In daily operations, INDAMS allows identifying unresponsive or dead nodes in the field, which are disconnected from the topology graph. Furthermore, continuous monitoring of voltage levels and health statistics help with better planning of maintenance visits, node replacements, and reconfigurations. Figure 7.17 shows the status of the ASWP testbed with 84 nodes color coded based on their battery level.

7.7 Data Functions

INDAMS stores all data collected from WSN testbeds in a centralized database, which acts as another client for the data handler component described earlier. These centralized data facilitate a new set of functions that act directly on stored data, without going through INDAMS lower layers. We have named these functions *data functions*. They are defined by WSN agents in INDAMS, as any other function, and user permissions are assigned in the same way.

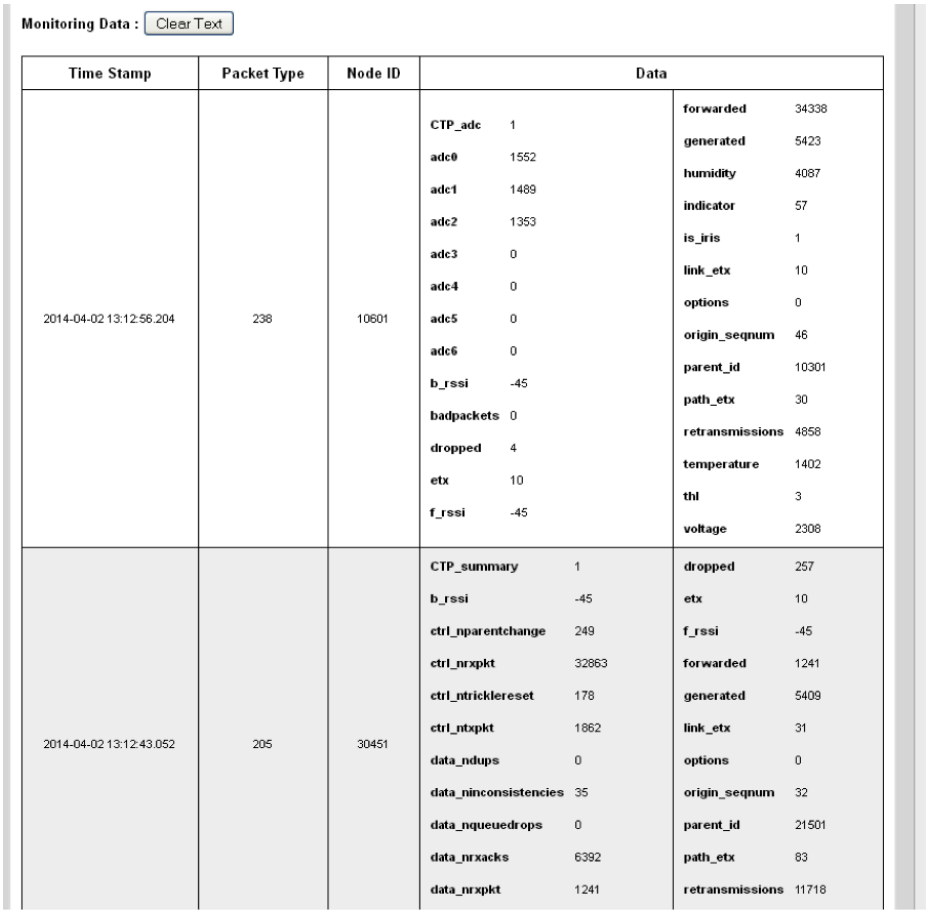


Figure 7.16. Data monitoring for the ASWP testbed.

The most basic data function is querying and exporting data from the database. INDAMS allows end users to specify date ranges, node IDs, and data types for performing a database query, which can be visualized in the web interface or exported to a file. In the same way, more complex queries can be implemented such as requesting the last received packet for a group of nodes. Figure 7.18 and Figure 7.19 show the web interface for these data functions.

7.7.1 Data Indicators

WSN applications at the ASWP testbed (i.e., XMesh-based or TinyOS-based WSN applications) include network health information that can be monitored from

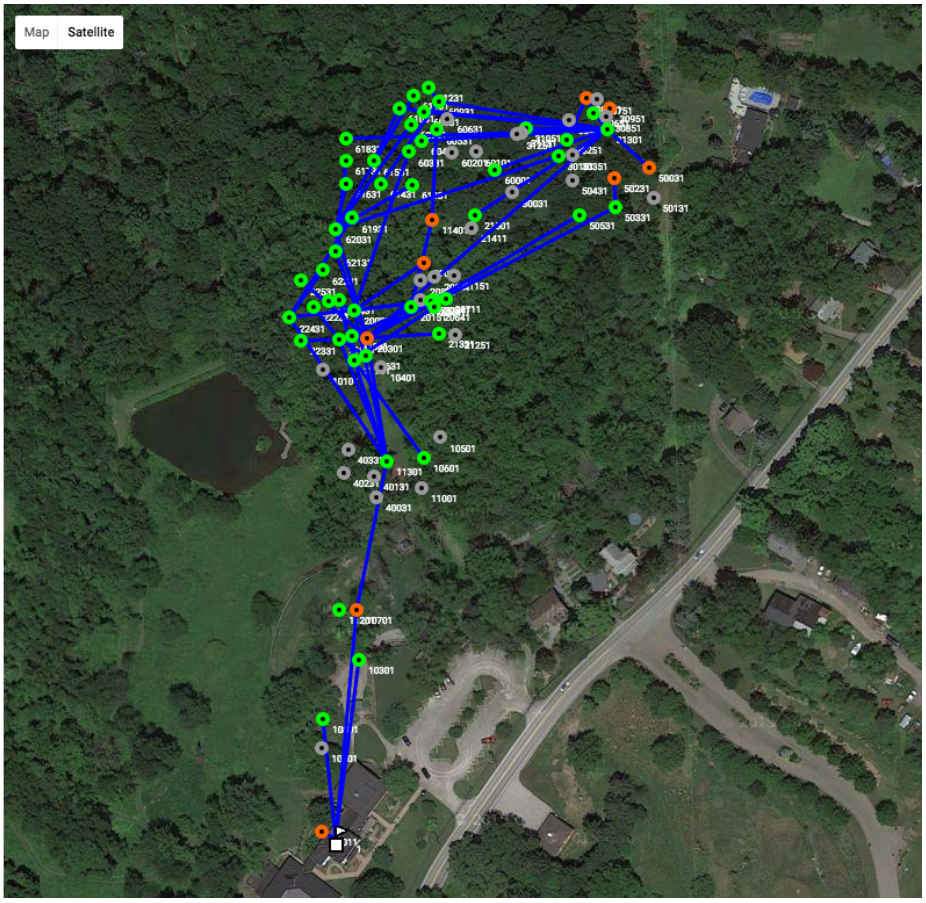


Figure 7.17. Topology monitoring and status of nodes at the ASWP testbed with 84 nodes. Node colors indicate their batteries levels: charged (green), depleted (gray), and close to be depleted (orange).

the data stream received in the data collection function. While this information provides valuable instant information, it is also important to understand the behavior of these variables over time. INDAMS defines a data indicator function, which helps visualizing these characteristics and trends in different time periods.

Indicators are defined from the health and instrumentation information provided by the WSN application and their behavior can be evaluated hourly, daily, or monthly. Unlike the data monitoring function, which subscribes to a continuous data stream; data indicators act directly on the database and the information can be updated based on the desired frequency.

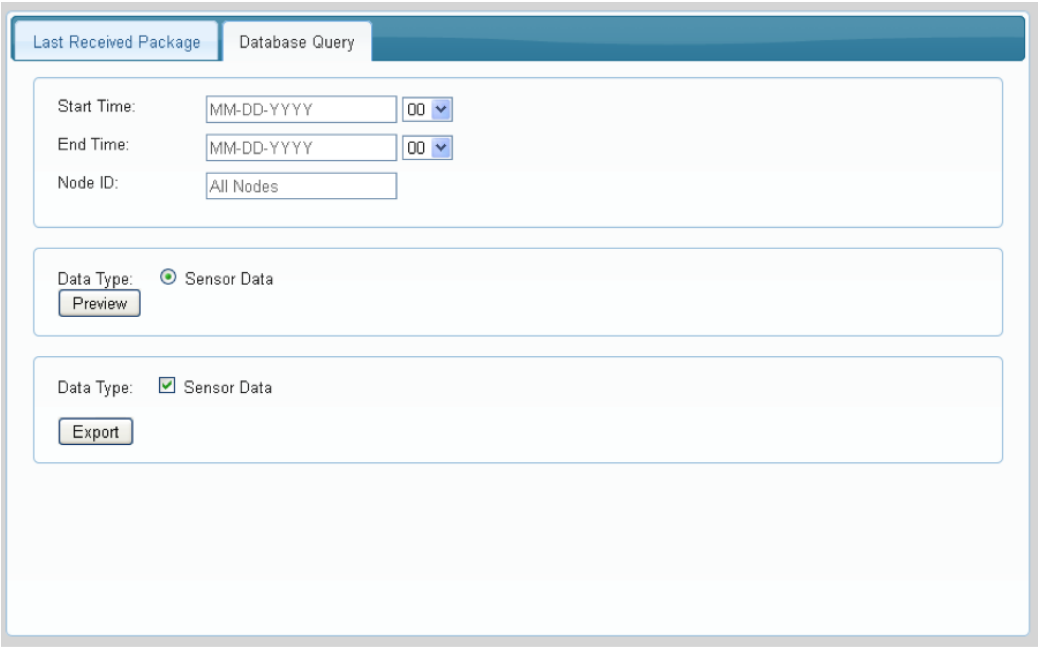


Figure 7.18. Web interface of the database query and export function in INDAMS.

The implementation of this function decouples the data processing from data visualization facilitating access for multiple users, knowing that data processing for some indicators may have a considerable complexity. In this way, data processing for each indicator is performed only once for each WSN agent and the results are stored in INDAMS central database. Then, when end users access the data indicator function, it only needs to query and plot the persisted results. An example of the web interface for the data indicator function is shown in Figure 7.20.

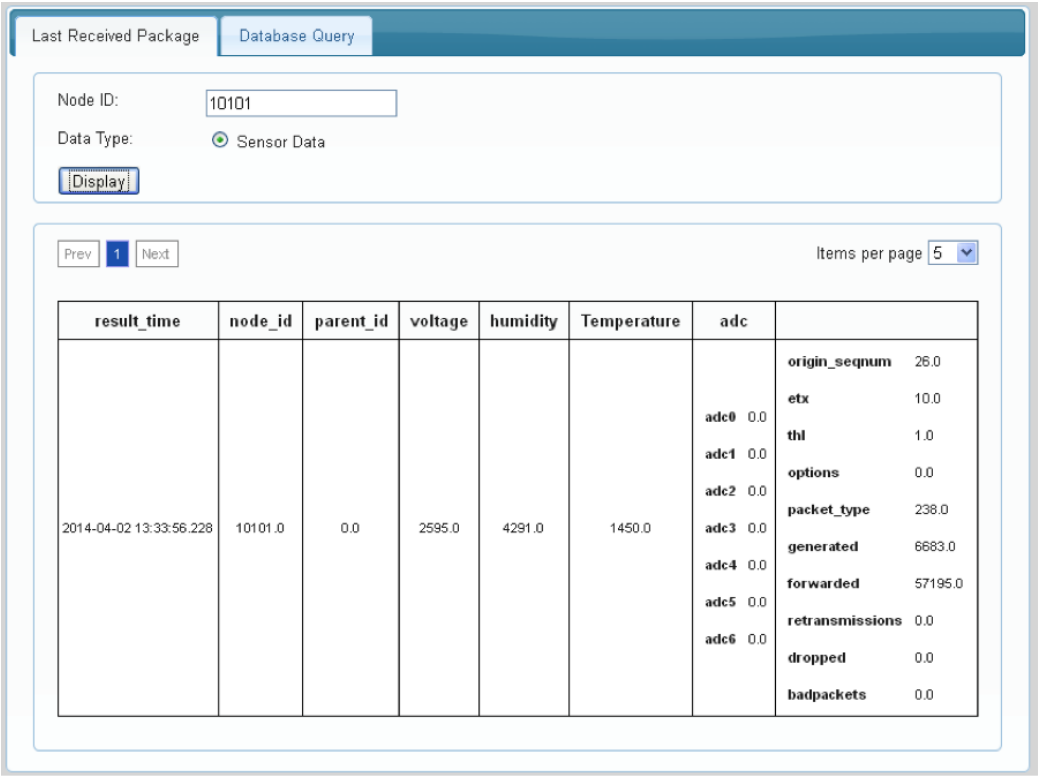


Figure 7.19. Web interface of the last received packet function in INDAMS.

Figure 7.20. Web interface for the data indicator function in INDAMS.

CHAPTER 8. SUMMARY AND FUTURE WORK

8.1 Summary

In this dissertation, we address the problem of energy efficiency in resource constrained and heterogeneous WSNs for data collection applications in real-world scenarios from three different perspectives: network routing, node energy profiles, and network management.

We present Energy Efficient Routing (EER), a new routing approach for energy-constrained data collection applications in multi-hop WSNs. Our approach introduces the concept of parent set for energy efficiency and balance in WSN routing, exploiting the redundancy offered by the network topology and leveraging on suboptimal and randomized routing alternatives in a controlled way. These route alternatives reduce the data traffic load on critical nodes, while maintaining high reliability in the network. The proposed EER provides a new diagnosis mechanism for network topology redundancy. In addition, EER can be implemented into any cost-based routing protocol, while remaining independent of the MAC layer. We demonstrate its implementation into CTP, which forms the new routing protocol CTP+EER. An analytical performance model is presented to define the redundancy conditions of the network topology that guarantee CTP+EER to improve the energy efficiency at the routing layer compared to CTP.

Our evaluation shows that CTP+EER overcomes the energy efficiency issues of traditional cost-based routing protocols and the reliability issues of state-of-the-art opportunistic routing protocols. In this way, CTP+EER defines a middle ground between sender-based and opportunistic routing, which combines high reliability and energy efficiency. CTP+EER achieved average PRRs over 99% in our testbed experiments and simulation scenarios, and at the same time, improved the maximum

transmission cost ranging from 11% to 59%. This energy efficiency of the routing layer resulted in the reductions of the maximum duty cycle ranging from 7% to 35%, when using the same asynchronous LPL configuration. Such high reliability and improvement of the network lifetime make CTP+EER very suitable for data collection applications in real-world energy-constrained WSN deployments, as we show it in a case study of the deployment of CTP+EER in the ASWP testbed, a real-world outdoor WSN testbed for environmental monitoring.

Our work on ERR is not only complimentary to other cost-based WSN routing protocols, but also to other energy-efficient MAC layer implementations, to further extend the network lifetime of practical WSN deployments.

The effect of the MAC layer on the network energy efficiency is studied based on the nodes energy consumption profile. Energy profiles are based on health and instrumentation data collected from WSN deployments and electric current measurements for various basic communication and sampling activities, taken from WSN nodes deployed in a laboratory setting. The proposed approach is applied to nodes deployed in the ASWP testbed. Relay and regular nodes were selected from multiple locations of the testbed and with different traffic conditions. Results reveal significant differences in the energy consumed by regular nodes compared to relay nodes, mainly due to (1) the energy consumption of external sensor sampling, and (2) the MDA300 acquisition board driver, which increases the current consumed by motes while sleeping. As relay nodes were configured to disable components related to the ADCs on MDA300 acquisition board, they have more energy-efficient operations. Relay nodes were found to be more sensitive to any changes in network traffic dynamics. Variations on outgoing packet transmissions (i.e., data transmissions, forwarded packets, re-transmissions and control packet transmissions) could account for higher percentages of the energy consumption due to the higher effort required from sending nodes in asynchronous LPL of TinyOS. This effect can be reduced for nodes located next to the sink node/base station by configuring the sink node not to sleep, while keeping the LPL preamble for packet transmissions.

Duty cycles computed based on the motes transceiver active time were compared with their energy profiles, and we found that the duty cycle alone does not reflect the real node energy consumption because of the effect of other hardware components (i.e., acquisition board and external sensors), and therefore energy profiles must also be considered for the design and implementation of energy efficient WSN applications and hardware platforms.

For data collection WSN deployments, network dynamics not only affect the performance of the WSN application, but also introduces high maintenance costs, i.e., replacing mote batteries. Our work has shown that WSN applications can be profiled in terms of energy consumption and we can identify the states of the application that should be improved to increase energy efficiency.

Finally, we presented our design and implementation of an integrated network and data management system for heterogeneous WSNs (INDAMS). INDAMS systematically supports heterogeneous WSNs with a unified management system, while separating the WSN management functions from WSN applications, and providing an easily accessible web interface for management functionalities. The system defines a five-layer architecture that provides the required levels of abstraction to handle platform/technology heterogeneity, application heterogeneity, and system scalability.

In order to handle multiple heterogeneous WSNs simultaneously, a core part of INDAMS is the agent-server communication within the UGL. The initial implementation proposed an agent-server protocol, which was later improved using a UG web service. The general structure of the agent-server protocol and the UG web service are presented and specific implementations are shown.

WSN abstraction is achieved by agents, which are responsible for implementing technology specific functions, interacting with WSN gateways. The agent implementation is presented for XServe and a generic TinyOS-based WSN gateway. Both implementations have been tested in the ASWP testbed supporting deployment, network monitoring, and maintenance operations.

The data handler is another major component of INDAMS. We adopted the concept of clients for generalizing the operation of this component, not only to users, but also to other applications and functions that may have similar requirements. The data handler is implemented based on the publisher/subscriber approach, and clients are handled as event listeners.

INDAMS supports user access control, allowing end users to access only relevant functions provided by their WSN agents through the web interface. WSN agents registered in INDAMS define provided functions in two high-level categories. The first category includes functions that require using the UGL, either request/response or continuous data functions. The second category corresponds to functions which do not need to use the UGL and can access INDAMS central database directly. These functions are defined as data functions. It was shown how all these functions support end-user operations of WSN deployment, network monitoring, and maintenance, by providing different options to access and visualize collected data. In addition, by facilitating the collection and processing of health and instrumentation data from WSNs, INDAMS enables additional functions such as the network redundancy diagnosis introduced by EER, as well as the energy profile calculations. This information is available to network administrators who can take appropriate actions as needed.

8.2 Future Work

EER revealed that with the proper definition of the parent set, a sender based approach can improve the reliability and energy efficiency compared to state-of-the-art receiver based approaches (i.e., opportunistic routing). EER can still be further improved by exploring new mechanisms for member selection of the parent set, which is currently done using a uniform distribution. This member selection from the parent set could be extended to define a new probability distribution based on additional traffic information to increase the probability of selecting members of the parent set with lower data traffic.

Another approach could be to combine the strength of receiver based approaches with EER. In a modified receiver based approach implementing EER, the parent set could be defined at the sender node and the member selection performed by receiving nodes. This approach would include the advantages from both mechanisms, exploiting faster and efficient data packet transmissions of opportunistic routing, and the proper parent set definition from EER.

Another future work would be to explore energy efficient MAC layer implementations that optimize the load balancing achieved by EER. Time synchronization and scheduling reduce preamble times and idle listing, further increasing the benefits of reducing the data traffic load processed in critical nodes.

Our energy profiles revealed the influence of external sensors in nodes energy consumption. Improved hardware and driver designs could reduce the constant energy consumption experienced by WSN nodes when ADCs are enabled. In addition, new approaches to combine information obtained from energy profiles with EER could also be explored, reducing the probability of high data traffic loads in nodes where other tasks are consuming significant energy.

Finally, INDAMS could be extended to provide new data functions for data analysis of sensor data, where sensor calibration plays a critical role in the quality of collected data. Furthermore, INDAMS could be implemented as a cloud-based service to facilitate the integration of new WSN deployments.

REFERENCES

REFERENCES

[1] David Culler, Deborah Estrin, and Mani Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, (8):41–49, 2004.

[2] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.

[3] Chiara Buratti, Andrea Conti, Davide Dardari, and Roberto Verdone. An overview on wireless sensor networks technology and evolution. *Sensors*, 9(9):6869–6896, 2009.

[4] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, and Eric Brewer. TinyOS: An operating system for sensor networks. *Ambient Intelligence*, pages 115–148, 2005.

[5] Chee-Yee Chong and Srikanta P Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[6] Guillermo Barrenetxea, Franois Ingelrest, Gunnar Schaefer, Martin Vetterli, Olivier Couach, and Marc Parlange. Sensorscope: Out-of-the-box environmental monitoring. *In the International Conference on Information Processing in Sensor Networks (IPSN)*, pages 332–343, 2008.

[7] Branko Kerkez, Steven D Glaser, Roger C Bales, and Matthew W Meadows. Design and performance of a wireless sensor network for catchmentscale snow and soil moisture measurements. *Water Resources Research*, 48(9), 2012.

[8] Christian Skalka and Jeffrey Frolik. Snowcloud: A complete data gathering system for snow hydrology research. *Real-World Wireless Sensor Networks*, pages 3–14, 2014.

[9] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. *In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 214–226, 2004.

[10] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, and David Gay. A macroscope in the redwoods. *In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pages 51–63, 2005.

[11] Xufei Mao, Xin Miao, Yuan He, Xiang-Yang Li, and Yunhao Liu. CitySee: Urban CO 2 monitoring with sensors. *In Proceedings of IEEE INFOCOM*, pages 1611–1619, 2012.

[12] Yunhao Liu, Yuan He, Mo Li, Jiliang Wang, Kebin Liu, and Xiangyang Li. Does wireless sensor network scale? A measurement study on GreenOrbs. *IEEE Transactions on Parallel and Distributed Systems*, 24(10):1983–1993, 2013.

[13] Md Zakirul Alam Bhuiyan, Guojun Wang, Jiannong Cao, and Jie Wu. Sensor placement with multiple objectives for structural health monitoring. *ACM Transactions on Sensor Networks (TOSN)*, 10(4):68, 2014.

[14] Nicolas Burri, Pascal Von Rickenbach, and Roger Wattenhofer. Dozer: Ultra-low power data gathering in sensor networks. *In the 6th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 450–459, 2007.

[15] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. Low-power wireless bus. *In Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 1–14, 2012.

[16] David Gay, Philip Levis, Robert Von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesC language: A holistic approach to networked embedded systems. *In ACM Sigplan Notices*, 38(5):1–11, 2003.

[17] Adam Dunkels, Bjrn Grnvall, and Thiemo Voigt. Contiki: A lightweight and flexible operating system for tiny networked sensors. *In the 29th Annual IEEE International Conference on Local Computer Networks (LCN)*, pages 455–462, 2004.

[18] Emmanuel Baccelli, Oliver Hahm, Matthias Whlisch, Mesut Gunes, and Thomas Schmidt. RIOT: One OS to rule them all in the IoT. Research Report RR-8176, INRIA, 2012.

[19] Thomas Watteyne, Xavier Vilajosana, Branko Kerkez, Fabien Chraim, Kevin Weekly, Qin Wang, Steven Glaser, and Kris Pister. OpenWSN: A standards-based lowpower wireless development environment. *Transactions on Emerging Telecommunications Technologies*, 23(5):480–493, 2012.

[20] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. *In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14, 2009.

[21] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, Maria Kazandjieva, David Moss, and Philip Levis. CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 10(1):16, 2013.

[22] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. *In Proceedings of the 1st Symposium on Networked Systems Design and Implementation*, pages 2–2, 2004.

[23] Douglas SJ De Couto. High-throughput routing for multi-hop wireless networks. PhD Thesis, Massachusetts Institute of Technology, 2004.

[24] Ugo Colesanti and Silvia Santini. The collection tree protocol for the Castalia wireless sensor networks simulator. Technical report, ETH Zurich, Zurich, Switzerland, 2011.

[25] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. *In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 14–27, 2003.

[26] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Levis. Four-bit wireless link estimation. *In Hot Topics in Networks (HotNets)*, 2007.

[27] Matthew Tancreti, Vinaitheerthan Sundaram, Saurabh Bagchi, and Patrick Eugster. TARDIS: Software-only system-level record and replay in wireless sensor networks. *In Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 286–297, 2015.

[28] David Moss, Jonathan Hui, and Kevin Klues. TEP 105: Low power listening, 2008. `http://www.tinyos.net/tinyos-2.x/doc/pdf/tep105.pdf` (Accessed 2016-04-01).

[29] Mo Sha, Gregory Hackmann, and Chenyang Lu. Energy-efficient low power listening for wireless sensor networks in noisy environments. *In Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, pages 277–288, 2013.

[30] David Moss and Philip Levis. BoX-MACs: Exploiting physical and link layer boundaries in low-power networking. Technical report, Stanford University, 2008.

[31] Tyler Davis. Environmental monitoring through wireless sensor networks. PhD Thesis, University of Pittsburgh, 2013.

[32] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. Motelab: A wireless sensor network testbed. *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, page 68, 2005.

[33] Emre Ertin, Anish Arora, Rajiv Ramnath, Vinayak Naik, Sandip Bapat, Vinod Kulathumani, Mukundan Sridharan, Hongwei Zhang, Hui Cao, and Mikhail Nesterenko. Kansei: A testbed for sensing at scale. *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pages 399–406, 2006.

[34] Manjunath Doddavenkatappa, Mun Choon Chan, and Akkihebbal L Ananda. Indriya: A low-cost, 3D wireless sensor network testbed. *Testbeds and Research Infrastructure. Development of Networks and Communities*, pages 302–316, 2012.

[35] Clment Burin Des Rosiers, Guillaume Chelius, Eric Fleury, Antoine Fraboulet, Antoine Gallais, Nathalie Mitton, and Thomas Nol. Senslab very large scale open wireless sensor network testbed. *In the 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCOM)*, 2011.

[36] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, pages 153–166, 2013.

[37] Tian He, Sudha Krishnamurthy, Liqian Luo, Ting Yan, Lin Gu, Radu Stoleru, Gang Zhou, Qing Cao, Pascal Vicaire, and John A Stankovic. VigilNet: An integrated sensor network system for energy-efficient surveillance. *ACM Transactions on Sensor Networks (TOSN)*, 2(1):1–38, 2006.

[38] Sandip Bapat, Vinodkrishnan Kulathumani, and Anish Arora. Analyzing the yield of exscal, a large-scale wireless sensor network experiment. *In the 13th IEEE International Conference on Network Protocols (ICNP)*, page 10 pp., 2005.

[39] Prabal Dutta, Jonathan Hui, Jaein Jeong, Sukun Kim, Cory Sharp, Jay Taneja, Gilman Tolle, Kamin Whitehouse, and David Culler. Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments. *In Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pages 407–415, 2006.

[40] Razvan Musaloiu-e, Andreas Terzis, Katalin Szlavecz, Alex Szalay, Joshua Cogan, and Jim Gray. Life under your feet: A wireless soil ecology sensor network. *In the 3rd Workshop on Embedded Networked Sensors (EmNets)*, 2006.

[41] MEMSIC. XMesh user manual. Revision A, 2010. `http://www.memsic.com/userfiles/files/User-Manuals/xmesh-user-manual-7430-0108-02_a-t.pdf` (Accessed 2016-04-01).

[42] MEMSIC. Moteview user manual. Revision D, 2012. `http://www.memsic.com/userfiles/files/User-Manuals/moteview-users-manual.pdf` (Accessed 2016-04-01).

[43] T. Winer, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander. RPL: IPv6 routing protocol for low-power and lossy networks, 2012. `https://datatracker.ietf.org/doc/rfc6550/` (Accessed 2016-04-01).

[44] Olaf Landsiedel, Euhanna Ghadimi, Simon Duquennoy, and Mikael Johansson. Low power, low delay: Opportunistic routing meets duty cycling. *In the 11th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 185–196, 2012.

[45] Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari, and Omprakash Gnawali. Routing without routes: The backpressure collection protocol. *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 279–290, 2010.

[46] Muhammad Hamad Alizai, Olaf Landsiedel, J gila Bitsch Link, Stefan Gtz, and Klaus Wehrle. Bursty traffic over bursty links. *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84, 2009.

[47] Daniele Puccinelli and Martin Haenggi. Reliable data delivery in large-scale low-power sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 6(4):28, 2010.

[48] Daniele Puccinelli, Silvia Giordano, Marco Zuniga, and Pedro Jos Marrn. Broadcast-free collection protocol. *In Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 29–42, 2012.

[49] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. ReInForM: Reliable information forwarding using multiple paths in sensor networks. *In the 28th Annual IEEE International Conference on Local Computer Networks (LCN)*, pages 406–415, 2003.

[50] Wenjing Lou. An efficient N-to-1 multipath routing protocol in wireless sensor networks. *In the IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 8 pp.–672, 2005.

[51] Hossam Hassanein and Jing Luo. Reliable energy aware routing in wireless sensor networks. *In the 2nd IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS)*, pages 54–64, 2006.

[52] Antoine B Bagula and Kuzamunu G Mazandu. Energy constrained multipath routing in wireless sensor networks. *Ubiquitous Intelligence and Computing*, pages 453–467, 2008.

[53] Mathieu Michel, Simon Duquennoy, Bruno Quoitin, and Thiemo Voigt. Load-balanced data collection through opportunistic routing. *In the International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 62–70, 2015.

[54] Farruh Ishmanov, Aamir Saeed Malik, and Sung Won Kim. Energy consumption balancing (ECB) issues and mechanisms in wireless sensor networks (WSNs): A comprehensive overview. *European Transactions on Telecommunications*, 22(4):151–167, 2011.

[55] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, page 10 pp. vol. 2, 2000.

[56] Chieh-Yih Wan, Shane B Eisenman, Andrew T Campbell, and Jon Crowcroft. Overload traffic management for sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 3(4):18, 2007.

[57] Rahul C Shah and Jan M Rabaey. Energy aware routing for low energy ad hoc sensor networks. *In the IEEE Wireless Communications and Networking Conference (WCNC)*, 1:350–355, 2002.

[58] Chang-Soo Ok, Seokcheon Lee, Prasenjit Mitra, and Soundar Kumara. Distributed energy balanced routing for wireless sensor networks. *Computers and Industrial Engineering*, 57(1):125–135, 2009.

[59] Hui Tian, Hong Shen, and Teruo Matsuzawa. Randomwalk routing for wireless sensor networks. *In the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 196–200, 2005.

[60] Sergio D Servetto and Guillermo Barrenechea. Constrained random walks on random graphs: Routing algorithms for large scale wireless sensor networks. *In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 12–21, 2002.

[61] Xavier Vilajosana, Jordi Llosa, Jose Carlos Pacho, Ignasi Vilajosana, Angel A Juan, Jose Lopez Vicario, and Antoni Morell. Zero: Probabilistic routing for deploy and forget wireless sensor networks. *Sensors*, 10(10):8920–8937, 2010.

[62] Rahim Kacimi, Riadh Dhaou, and Andre-Luc Beylot. Load balancing techniques for lifetime maximizing in wireless sensor networks. *Ad Hoc Networks*, 11(8):2172–2186, 2013.

[63] Li Shancang, Zhao Shanshan, Wang Xinheng, Zhang Kewang, and Li Ling. Adaptive and secure load-balancing routing protocol for service-oriented wireless sensor networks. *IEEE Systems Journal*, 8(3):858–867, 2014.

[64] Fatma Bouabdallah, Nizar Bouabdallah, and Raouf Boutaba. On balancing energy consumption in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 58(6):2909–2924, 2009.

[65] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with Cooja. *In Proceedings 31st IEEE Conference on Local Computer Networks*, pages 641–648, 2006.

[66] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. *In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 126–137, 2003.

[67] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. *in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 188–200, 2004.

[68] Olaf Landsiedel, Klaus Wehrle, and Stefan Gtz. Accurate prediction of power consumption in sensor networks. *In the 2nd Workshop on Embedded Networked Sensors*, 2005.

[69] Ben L Titzer, Daniel K Lee, and Jens Palsberg. Avrora: Scalable sensor network simulation with precise timing. *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, page 67, 2005.

[70] Jingyao Zhang, Srikrishna Iyer, Patrick Schaumont, and Yaling Yang. Simulating power/energy consumption of sensor nodes with flexible hardware in wireless networks. *In the 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 112–120, 2012.

[71] Jingyao Zhang, Yi Tang, Sachin Hirve, Srikrishna Iyer, Patrick Schaumont, and Yaling Yang. A software-hardware emulator for sensor networks. *In the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 440–448, 2011.

[72] Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. Software-based on-line energy estimation for sensor nodes. *In Proceedings of the 4th Workshop on Embedded Networked Sensors*, pages 28–32, 2007.

[73] Xiaofan Jiang, Prabal Dutta, David Culler, and Ion Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 186–195, 2007.

[74] Matthias Woehrle, Jan Beutel, Roman Lim, Mustafa Yuecel, and Lothar Thiele. Power monitoring and testing in wireless sensor network development. *In the Workshop on Energy in Wireless Sensor Networks (WEWSN)*, 2008.

[75] Prabal Dutta, Mark Feldmeier, Joseph Paradiso, and David Culler. Energy metering for free: Augmenting switching regulators for real-time monitoring. *In the International Conference on Information Processing in Sensor Networks (IPSN)*, pages 283–294, 2008.

[76] Decagon Devices. EC-5 soil moisture sensor. Version 2, 2012. `http://www.decagon.com/products/soils/volumetric-water-content-sensors/ec-5-soil-moisture-small-area-of-influence/` (Accessed 2016-04-01).

[77] Decagon Devices. MPS-1 dielectric water potential sensor. Version 3, 2009. `http://www.decagon.com/products/soils/water-potential/` (Accessed 2016-04-01).

[78] Tyler W Davis, Chen-Min Kuo, Xu Liang, and Pao-Shan Yu. Sap flow sensors: Construction, quality control and comparison. *Sensors*, 12(1):954–971, 2012.

[79] Mihai Marin-Perianu, Nirvana Meratnia, Paul Havinga, Luciana Moreira S de Souza, Jens Mller, Patrik Spie, Stephan Haller, Till Riedel, Christian Decker, and Guido Stromberg. Decentralized enterprise systems: A multiplatform wireless sensor network approach. *IEEE Wireless Communications*, 14(6):57–66, 2007.

[80] Bhawana Parbat, AK Dwivedi, and OP Vyas. Data visualization tools for WSNs: A glimpse. *International Journal of Computer Applications*, 2(1):14–20, 2010.

[81] Gilman Tolle and David E Culler. Design of an application-cooperative management system for wireless sensor networks. *In proceedings of EWSN*, 5:121–132, 2005.

[82] D Kim, H Song, Kangwoo Lee, and Jongwoo Sung. UPnP-based sensor network management architecture. *In the 2nd International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, 2005.

[83] Fabrcio A Silva, Linnyer Beatrys Ruiz, Thais Regina M Braga, Jos Marcos S Nogueira, and Antonio Alfredo Ferreira Loureiro. Defining a wireless sensor network management protocol. *In the Latin American Network Operations and Management Symposium (LANOMS)*, pages 39–50, 2005.

[84] Linnyer Beatrys Ruiz, Jos Marcos Nogueira, and Antonio AF Loureiro. Manna: A management architecture for wireless sensor networks. *Communications Magazine*, 41(2):116–125, 2003.

[85] Wei Liu, Yanchao Zhang, Wenjing Lou, and Yuguang Fang. Managing wireless sensor networks with supply chain strategy. *In the 1st International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE)*, pages 59–66, 2004.

[86] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver. Winms: Wireless sensor network-management system, an adaptive policy-based management for wireless sensor networks. Technical report, The University of Western Australia, 2006.

[87] Raja Jurdak, Antonio G Ruzzelli, Alessio Barbirato, and Samuel Boivineau. Octopus: Monitoring, visualization, and control of sensor networks. *Wireless Communications and Mobile Computing*, 11(8):1073–1091, 2011.

[88] Biljana Stojkoska and Danco Davcev. Web interface for habitat monitoring using wireless sensor network. *In the 5th International Conference on Wireless and Mobile Communications*, pages 157–162, 2009.

[89] Ioannis Chatzigiannakis, Georgios Mylonas, and Sotiris Nikoletseas. jWebDust: A java-based generic application environment for wireless sensor networks. *Distributed Computing in Sensor Systems*, pages 376–386, 2005.

[90] MEMSIC. Moteworks getting started guide. Revision G, 2012. `http://www.memsic.com/userfiles/files/User-Manuals/moteworks-getting-started-guide.pdf` (Accessed 2016-04-01).

[91] MEMSIC. XServe users manual. Revision A, 2010. `http://www.memsic.com/userfiles/files/User-Manuals/xserve_users_manual-7430-0111-02_a-t.pdf` (Accessed 2016-04-01).

[92] Newlyn Erratt and Yao Liang. The design and implementation of a general WSN gateway for data collection. *In the IEEE Wireless Communications and Networking Conference (WCNC)*, pages 4392–4397, 2013.

APPENDIX

APPENDIX. PACKET FORMAT AND APPLICATION FOR CTP+EER

CTP+EER maintains the same structure for routing and data packets as described in Chapter 2. Then, to enable the network diagnosis in EER, a minimum instrumentation needs to be appended to the header of data packets.

Figure A.1 shows the basic data packet structure in CTP+EER for network diagnosis. It includes the size of the parent set and the ID of the parent node used to forward the current data packet. Any additional instrumentation can still be appended to this modified header, or included in the application payload.

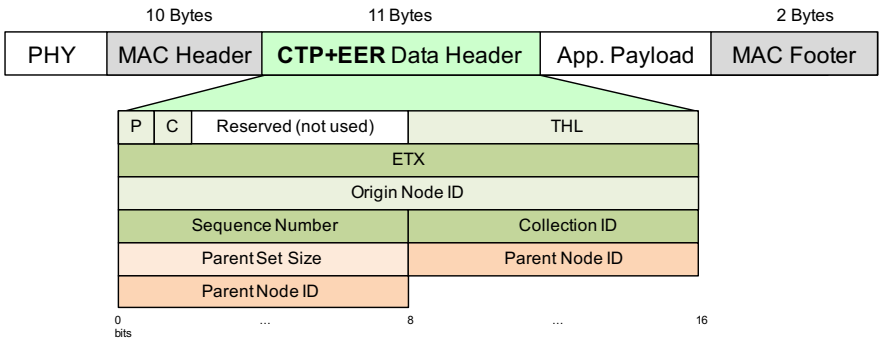


Figure A.1. CTP+EER basic data packet structure for network diagnosis.

The implementation of CTP+EER maintains the interface of CTP to the application layer. It uses the components `CollectionC` and `CollectionSenderC` as shown bellow, and the application components can wire all interfaces related to the radio.

```
configuration MotesAppC {
}
implementation {
  ...
  components CollectionC as Collector;
  components new CollectionSenderC(0xee);
  ...
  MotesC.RadioControl -> ActiveMessageC;
  MotesC.RoutingControl -> Collector;
  MotesC.Send -> CollectionSenderC;
  MotesC.CtpInfo -> Collector;
  ...
}
```

Finally, the `Makefile` of the TinyOS application only needs to specify the maximum size of the parent set in CTP+EER for memory allocation as shown below.

```
...
# Maximum parent set size in CTP+EER
CFLAGS += -DMAX_PARENTSET_SIZE=5
...
```

VITA

VITA

Miguel Andrés Navarro Patiño

Education

- PhD, Computer Science. Purdue University, West Lafayette, Indiana, USA
- MS, Systems and Computing Engineering. Universidad de los Andes, Bogotá, Colombia
- BS, Electronics Engineering. Universidad de los Andes, Bogotá, Colombia

Research Interests

Internet of things, wireless sensor networks, cyber-physical systems, and machine learning.

PUBLICATIONS

PUBLICATIONS

1. Miguel Navarro, Diviyansh Bhatnagar, and Yao Liang. An integrated network and data management system for heterogeneous WSNs. *In the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*. Valencia, Spain. October, 2011

2. (Demo) Miguel Navarro, Diviyansh Bhatnagar, Rui Liu, and Yao Liang. Design and implementation of an integrated network and data management system for heterogeneous WSNs. *In the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*. Valencia, Spain. October, 2011

3. Tyler Davis, Xu Liang, Miguel Navarro, Diviyansh Bhatnagar, Yao Liang. An experimental study of WSN power efficiency: MICAz networks with XMesh. *International Journal of Distributed Sensor Networks*, vol. 2012, Article ID 358238, 2012

4. (Poster) Miguel Navarro, Tyler W. Davis, Yao Liang, Xu Liang. ASWP: A long-term WSN deployment for environmental monitoring. *In the 12th ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN)*. Philadelphia, USA. April, 2013

5. Miguel Navarro, Tyler W. Davis, Yao Liang, Xu Liang. A study of long-term WSN deployment for environmental monitoring. *In the 24th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. London, UK. September, 2013

6. Miguel Navarro, Yimei Li, Yao Liang. Energy profile for environmental monitoring wireless sensor networks. *In the IEEE Colombian Conference on Communications and Computing*. Bogotá, Colombia. June, 2014

7. (Poster) Xiaoyang Zhong, Miguel Navarro, German Villalba, Xu Liang, Yao Liang. MobileDeluge: A novel mobile code dissemination for wireless sensor networks. *In the IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*. Philadelphia, USA. October, 2014

8. Xiaoyang Zhong, Miguel Navarro, German Villalba, Xu Liang, Yao Liang. MobileDeluge: Mobile code dissemination for wireless sensor networks. *In the IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*. Philadelphia, USA. October, 2014

9. Miguel Navarro, Tyler W. Davis, German Villalba, Yimei Li, Xiaoyang Zhong, Newlyn Erratt, Xu Liang, Yao Liang. Towards long-term multi-hop WSN deployments for environmental monitoring: An experimental network evaluation. *Journal of Sensor and Actuator Networks*. DOI: 10.3390/jsan3040297. December, 2014

10. Miguel Navarro, Yao Liang. Efficient and balanced routing in energy-constrained wireless sensor networks for data collection. *In the International Conference on Embedded Wireless Systems and Networks (EWSN)*. TU Graz, Austria. February, 2016

11. Miguel Navarro, Yao Liang. Efficient and balanced routing in energy-constrained wireless sensor networks for data collection. *(Submitted IEEE Journal)*