ENERGY-EFFICIENT PROTOCOLS AND SYSTEMS FOR WIRELESS SENSOR NETWORKS AND SMART ENVIRONMENTS

by

GIACOMO GHIDINI

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2012

To my family.

ACKNOWLEDGMENTS

During my graduate studies, I have been very fortunate to receive the constant support of many great family, friends, and colleagues. Prof. Das has been a great supervisor, and I especially appreciate how he has always known when to motivate and push me, as well as when to let me search for the right path on my own. My Ph.D. committee members are experts in their research areas, and I am very proud to have discussed my research work with them. The Center for Research in Wireless Mobility and Networking is a great community of researchers, and I am thankful to its current and past members, as well as all the visitors who have shared their knowledge with us during seminars and meetings. The Department of Computer Science and Engineering at UT Arlington is a great place to work at, and its faculty and staff have always helped me sort out any issue related to my studies. Many other people have played important roles in my graduate studies, and I would like to especially thank Dr. Vipul Gupta of Oracle Labs. My family and friends deserve a lot of credit for always being there for me: cheering me up when I was going through hard times, as well as rejoicing at my achievements. It is thank to all these great people that I was able to achieve my research goals and graduate from the Ph.D. program.

November 12, 2012

ABSTRACT

ENERGY-EFFICIENT PROTOCOLS AND SYSTEMS FOR
WIRELESS SENSOR NETWORKS AND
SMART ENVIRONMENTS

GIACOMO GHIDINI

The University of Texas at Arlington, 2012

Supervising Professor: Sajal K. Das

In a wireless sensor network, small computing devices, called sensors, sense the surrounding environment and relay the sensed data to a base station over a multi-hop wireless network, eventually processing them en-route. Wireless sensor networks and other devices, such as smartphones, smart meters, and smart appliances, cooperate in smart environments to obtain information about the environment, and then use this information to improve the experience of the users. Since most of these systems rely on battery power, there is a need for energy-efficient solutions for their operation. The objective of this dissertation is to design algorithms and protocols to improve the energy efficiency of such systems, and validate them using mathematical analysis, software simulations, and testbed experiments.

In the first part of the dissertation, we look at two fundamental problems in wireless sensor networks: localization and duty cycling. In the area of localization, we describe a novel protocol for duty cycling wireless actor and sensor networks, and present a mathematical analysis based on the coupon collector's problem and the the-

ory of coverage processes, as well as simulation results. Our analysis and results show that the proposed protocol achieves the user-requested localization accuracy while maximizing the sleep time of sensor nodes. As far as duty cycling is concerned, we present novel Markov chain-based randomized schemes, and discuss the probabilistic analysis, as well as the experiments we conducted on Sun SPOT sensors. These results show that our proposed schemes reduce the sleep latency, while not affecting other performance metrics such as the energy efficiency, or vice versa.

In the second part of the dissertation, we shift our focus to smart environments, and present our research work on data fusion and visualization aimed to provide lay users with actionable information. We introduce a framework, called FuseViz, to leverage already existing data sources such as smartphones, online databases and services, and wireless sensor networks, while addressing the challenges posed by large, live, heterogeneous, and autonomous data streams. We demonstrate the concepts behind our framework with a case study in building energy efficiency, and introduce E^2Home, a Web-based application for this problem developed on top of the framework. Preliminary experiment results for the proposed E^2Home system not only show that the actionable information can be easily computed, but also demonstrate energy savings of about 10%. Finally, we conclude our dissertation with an overview of a system-level energy model, built using data from the above-mentioned sources, that can be tailored for each home, its location, and residents, and can help further minimize energy consumption.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

A wireless sensor network (WSN), depicted in Figure 1.1, consists of a set of sensor nodes, small battery-powered computing devices connected via a multi-hop wireless network [1, 2]. These nodes, also called sensors or motes, measure physical quantities of the surrounding environments using on-board sensors. Thanks to advances in micro-electronic mechanical systems (MEMS), there exist small form-factor sensors to measure a wide array of physical quantities: from temperature to humidity, from strain to electromagnetic field, to name a few. These analog data are then converted to digital values and relayed to the base station, or sink, along a multi-hop route formed by the motes in the wireless network. These data may be stored in memory at the source node, en route, or the base station, and processed by these nodes, in order to remove redundancy or noise, add error correction, or verify authenticity. Once the data reach the base station, they may be used to monitor the environment in which the WSN is deployed, build a model thereof, and make decisions as to what actions need be taken by actuators or humans. As such, WSNs have a wide range of applications from wildlife [3] and structural health monitoring [4] to traffic management [5] and industrial processes [5].

In a smart environment, advanced sensing, computing, and communications are employed to bring intelligence in the physical environment to the advantage of the users. Among the many technologies used in smart environments, wireless sensor networks play a key role due to their ability to sense potentially large areas at a fine

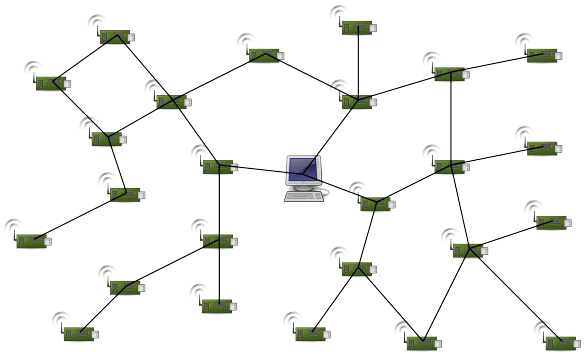


Figure 1.1. A wireless sensor network consisting of motes connected to a base station via a multi-hop wireless network.

temporal and spatial resolution, and make the (aggregated) information available to the users themselves, as well as artificial intelligence and control systems.

This dissertation presents our research work in the areas of wireless sensor networks and smart environments. Figure 1.2 offers a visual summary of the research work presented in this dissertation. Our research focuses mainly on the design and analysis of energy-efficient protocols and systems for WSNs and smart environments. Before presenting our research work, we survey state-of-the-art research in wireless sensor networks with a special focus on the communication stack and the standards developed by the Institute of Electrical and Electronics Engineers (IEEE) and the Internet Engineering Task Force (IETF) in Ch. 2.

In Ch. 3, we introduce an improved version of a localization protocol for wireless sensor and actor networks, and discuss its performance by means of our mathematical analysis based on results for the coupon collector's problem and the theory of coverage processes, and simulation results. The mathematical analysis, also validated by the simulation results, shows that the proposed algorithm can successfully localize the desired share of sensors in the network with minimum operations of these energy-constrained devices.

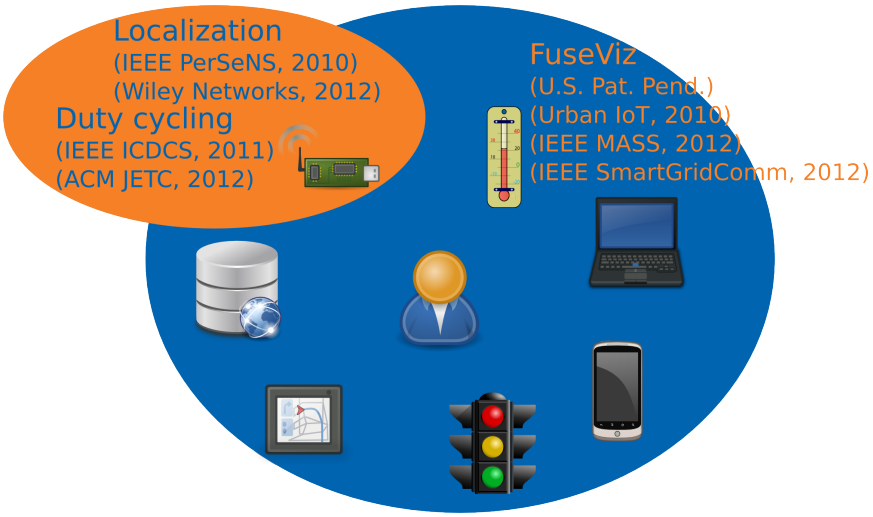


Figure 1.2. Scenario and research work.

In Ch. 4 and Ch. 5, we present our proposed Markov chain-based randomized schemes for increased duty cycling energy efficiency in wireless sensor networks, and discuss their performance using our analysis based on probability theory and experimental results for Sun SPOTs. We prove mathematically and demonstrate experimentally that the energy efficiency of randomized duty cycling schemes can be improved using our Markov chain-based solution while not negatively affecting other performance metrics such as the connection delay.

In Ch. 6, we move from wireless sensor networks to smart environments and look at the problem of leveraging present and future large, live, heterogeneous and independent data sources for acquiring actionable information for lay users. We demonstrate the feature of the proposed framework in E^{2}Home, a Web-based system for a case study in home energy efficiency. Initial experiments in homes show that the contextual information provided by the E^{2}Home system can help residents reduce electricity consumption by more than 10%. Finally, in Ch. 7 we draw our conclusions and suggest directions for future work.

CHAPTER 2

AN INTRODUCTION TO WIRELESS SENSOR NETWORKS AND THE COMMUNICATION STACK

Given their nature as a direct connection between the physical and cyber worlds, WSNs have potential applications in many diverse fields, and are one of the fundamental components to develop greener computing systems. In fact, one can imagine that most problems humankind is facing could be addressed in a more precise and (energy) efficient manner, if accurate data from the physical environment (be it a forest hosting an endangered species, a crop needing the appropriate quantity of water and fertilizers, or a freeway network often jammed by traffic) were available. So far, WSNs have been applied in several areas, including natural environments [6], rural areas [7], and urban domains [8]. Constraints on the motes due to the current technology, such as battery lifetime, size of the devices, and manufacturing costs, may temporarily delay the widespread application of WSNs to certain domains and scenarios. While WSNs are a necessary component towards more energy efficient systems in many domains, it is also important that the WSN protocols and applications be designed with energy efficiency as one of the driving objectives in order to make the overall system greener.

In this chapter, we aim to provide an overview of state-of-the-art algorithms and protocols for wireless sensor networks with a special focus on the communication stack[1]. In all distributed systems, the protocols for the communication stack play

[1]The research work presented in this chapter is under review for publication in Design Technologies for Green and Sustainable Computing Systems [9].

a vital role by enabling interoperability of different subsystems. In wireless sensor networks, the communication stack is even more important because it accounts for the largest share of energy consumption in many application scenarios, and thus well-designed, energy-efficient protocols have a strong impact on the overall lifetime of the WSN.

In the past decade, several organizations, including the Institute of Electrical and Electronics Engineers (IEEE) and the Internet Engineering Task Force (IETF), have defined standards for physical layer, MAC layer, and network layer. In particular, we observe a trend towards the adoption of IEEE 802.15.4 [10] for physical (PHY) and medium access control (MAC) layers, and IETF 6LoWPAN [11, 12] as the IPv6 protocol at the network layer. Standards for routing and ReSTful communication are also being proposed within the IETF. The IETF Routing over Low Power and Lossy Networks (RoLL) working group (WG) is developing the Routing Protocol for LLNs (RPL) [13], a standard for routing in WSNs, while the Constrained ReSTful Environments (CoRE) [14] is developing the Constrained Application Protocol (CoAP), a standard for ReSTful communication with WSNs. Figure 2.1 displays these protocols side-by-side with the corresponding ones already being used in the Internet.

We argue that it is important to provide an overview of the existing and proposed standards and their implementations not limiting the analysis to one layer, but rather discussing them as part of this developing communication stack for WSNs. To this extent, in our presentation of state-of-the-art solutions and proposed standards we attempt to bring to the foreground the interdependencies between different layers and the implications that design decisions at one layer have on the performance at other ones. Ultimately, our analysis of the communication stack is aimed to help researchers who are new to the area of WSN communications understand its overall

functioning, while also offering to more seasoned researchers an insight into protocols at different layers and the interplay among them.

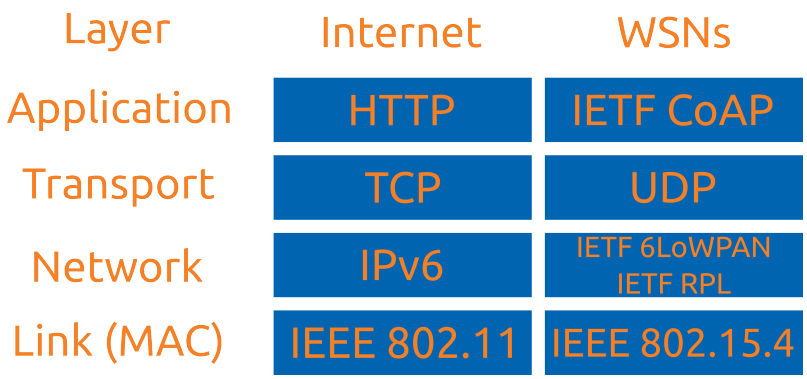


Figure 2.1. Comparison of the communication stacks used in the Internet and in WSNs.

The rest of the chapter is organized as follows. In Section 2.1, we review the different classes of MAC protocols, and describe the major features of IEEE 802.15.4. In Section 2.2, we present 6LoWPAN and RPL, respectively the standards for IPv6 and routing in WSNs, and survey implementations and results. In Section 2.3, we summarize the major features of CoAP, and then analyze recent evaluations of the protocol. Finally, we draw our conclusions in Section 3.4.

2.1 MAC Layer

In a WSN, the MAC layer plays a vital role as it enables the actual communication among nodes over a common medium. As such, a MAC protocol is often evaluated along several dimensions, including delay, throughput, and energy efficiency. However, constraints such as scarce battery capacity, limited computational power, and small memory size, make the design of MAC protocols that can perform well with respect to the performance metrics very difficult. Finally, the diversity of traffic

patterns generated by applications in diverse domains further complicate the design of a general, effective, and efficient MAC protocol for WSNs.

Likely because of the challenges discussed above, researchers have dedicated a lot of efforts to developing MAC protocols that meet all the requirements. As a results, dozens, if not hundreds, of MAC protocols have been proposed by the research community in the past 15 years. In the past, and to a certain extent, still today, authors tried to classify MAC protocols based on the technique that they employ to coordinate access to a common medium by multiple nodes. These classifications are often variations of the one considering reservation-based protocols, contention-based protocols, and hybrid solutions. In this classification, reservation-based protocols usually feature some form of time-division multiple access (TDMA) and/or frequency-division multiple access (FDMA), while contention-based protocols are built around ALOHA or carrier-sense multiple access (CSMA), and hybrid protocols feature a mix of the two. The requirement for knowledge of the topology and strict synchronization are among the major drawbacks of the first class of protocols (*i.e.,* reservation-based). Instead, the second class of protocols (*i.e.,* contention-based) does not require this information, but experiences degraded performance in case of heavy traffic load and high energy consumption per bit (*i.e.,* poor energy-efficiency) even in presence of light traffic loads.

2.1.1 MAC Protocol Classes

Recently, [15] proposes a classification of MAC protocols based on traffic patterns. The important assumption behind this classification is that the ultimate MAC protocol for WSNs with optimal performance for all traffic loads does not exist. Instead, there exist MAC protocols that are optimal for certain classes of WSN traffic.

The authors first summarize the causes of wasteful energy consumption at the MAC layer. In particular, they list: collisions, overhearing, overhead, and idle listening. Then they define three classes of traffic load: heavy, medium, and low. They observe how certain kinds of wasteful energy consumption are more likely to arise in presence of specific traffic loads. For instance, collisions are more common in case of heavy traffic, while idle listening is usually a cause of wasteful energy consumption in presence of light traffic. The authors then introduce three basic classes of MAC protocols: scheduled protocols, common active period-based protocols, and preamble sampling-based protocols. Hybrid protocols, such as IEEE 802.15.4, present features of different classes, and thus are grouped in a separate class.

2.1.1.1 Schedule-based Protocols

In a scheduled protocol like TSMP [16], medium access is controlled by a schedule in the time and/or frequency domains. In a first version, communication links for each pair of neighboring nodes can be scheduled. Figure 2.2 portrays the communication between all pairs of nodes in a clique of four sensors. Alternatively, simply the senders or the receivers can be scheduled. The first option performs very well in presence of heavy traffic loads, but brings about major overhead since all pairs of neighboring nodes must be scheduled a slot for communication. Overhead is reduced by scheduling only senders or receivers, but other sources of wasteful energy consumption become relevant. In the solution where senders are scheduled, all neighbors have to listen to each sender, because the message may be addressed to them, thus resulting in overhearing. If receivers are scheduled instead, collisions may occur, so that this variant is not as effective in case of heavy traffic loads.

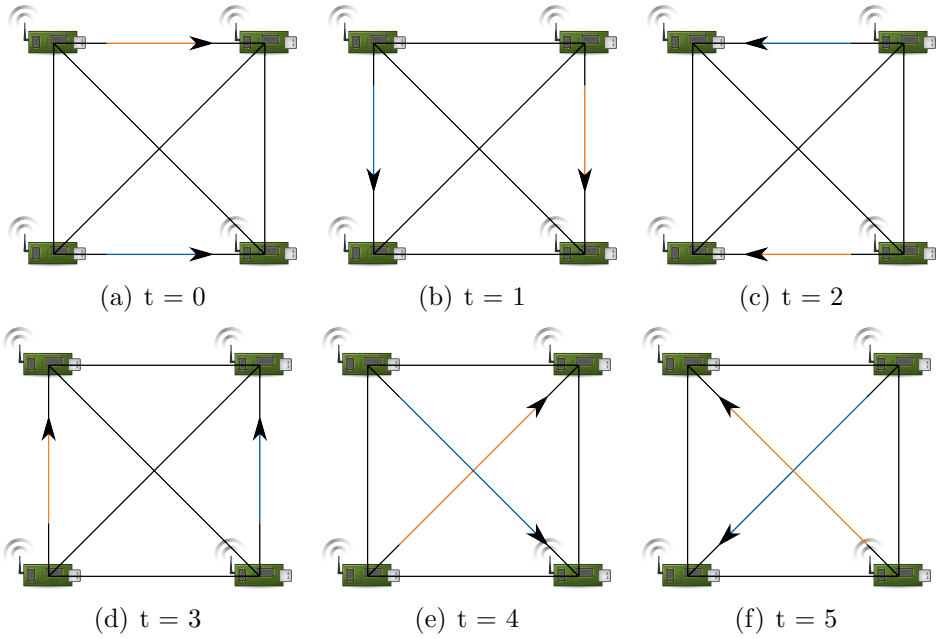


Figure 2.2. Sample scheduled MAC protocol. Black lines depict communication links; and orange and blue arrows depict scheduled communication on two different channels in (a) through (f) for time slots 0 through 5.

2.1.1.2 Common Active Period-based Protocols

The next class of MAC protocols is that of those based on common active periods and targeted to medium traffic loads such as SMAC [17]. As depicted in Figure 2.3, protocols in this class attempt to achieve a coarse synchronization between the active periods of neighboring nodes, so that they can communicate during these times, and turn off the radios at all other times. This is based on the assumption that a medium traffic load can be taken care of during a fraction of the node lifetime, and thus precious battery power can be saved by operating the radio only during these times. In the common active periods, nodes usually operate according to a contention-based mechanism such as CSMA to transmit their frames. In a common active period-based protocol, schedules are distributed so that all neighbors turn on their radios at the same time. As a result of this schedule distribution, clusters of nodes with the

same schedule can be formed. As depicted in Figure 2.3, in order to support inter-cluster communication, certain nodes are required to keep their radio on according to the union of all known schedules, and incur into higher energy consumption as a result.

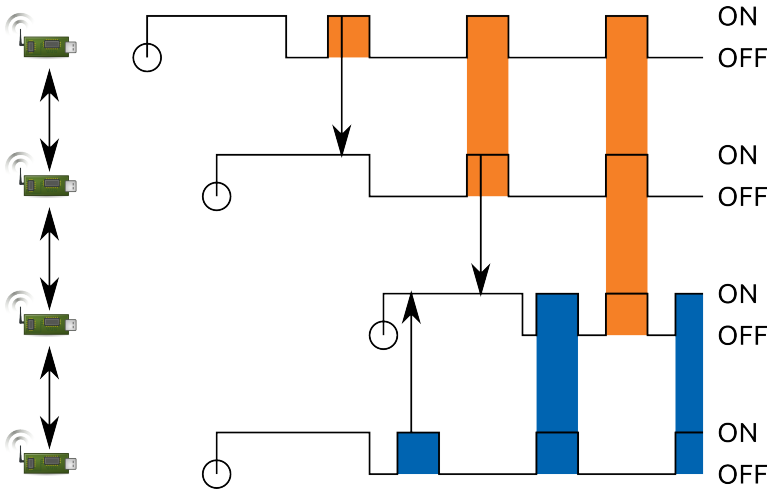


Figure 2.3. Example of schedule distribution in SMAC. Two different schedules (orange and blue) are distributed to nodes, and eventually border nodes have to accept both.

2.1.1.3 Preamble Sampling-based Protocols

In the third class of MAC protocols, targeted to low traffic loads, nodes use preamble sampling to synchronize communications. In a preamble sampling-based protocol such as the low-power listening (LPL) mechanism used in TinyOS and BMAC [18], the sender transmits a beacon to announce that it has a message to relay to a neighboring node. As depicted in Figure 2.4, all nodes periodically turn on their radios for brief periods of time, and sample the channel looking for these beacons. If such a beacon is received, then the node keeps its radio on and waits for the actual transmission from the sender. There exist several variations of this basic mode of

operations. For instance, synchronization information can be piggybacked on the beacon like in WiseMAC [19], so that the potential receivers do not have to keep their radios on, but rather can turn them off and then turn them on again at the time of communication as announced by the sender. In other protocols such as X-MAC [20], senders transmit the beacon in periodic short preambles instead of a single long one, so that receivers can acknowledge the reception of the preamble without waiting for its end. Finally, in a reversal of the original protocol, receivers may transmit the beacon to initiate transmission from the senders as it is the case in RICER [21].

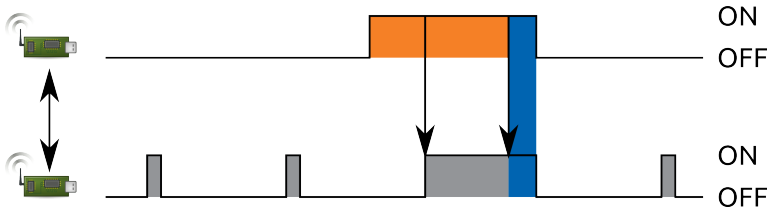


Figure 2.4. Example of preamble sampling-based MAC protocol.

2.1.1.4 Hybrid Protocols

Besides the three classes described above, there exist also hybrid MAC protocols. The objective of these protocols is to optimize performance across different traffic loads. To achieve this goal, hybrid protocols employ several techniques. For instance, they may rely on flexible MAC frame structure like IEEE 802.15.4, so that different modes of operation can be applied. In particular, the non-beacon mode of this standard protocol is basically a CSMA with collision avoidance (CSMA/CA). Instead, in the beacon-enabled mode so-called collision free period (CFPs) may be scheduled by the coordinator for specific nodes, while CSMA is still available for the rest of the time. Another solution is to blend a reservation-based protocol with a

contention-based mechanism. In a hybrid protocol like ZMAC [22], nodes operate according to CSMA whenever traffic load is light, but can set up a schedule and switch to TDMA when they observe heavier traffic. Finally, in protocols such as Funneling MAC [23], nodes can operate according to a contention-based mechanism if they are further away from the sink where traffic load is low, and use reservation-based techniques if they are in the surroundings of the sink, where convergecast traffic brings about heavier loads.

2.2 Network Layer

In the past several decades, the Internet has thrived also thank to the availability of IP (the Internet Protocol) across different devices and networks. There is widespread agreement within the WSN research community that WSNs and other constrained networks will fulfill their potential, if they can seamless interoperate with the Internet. In order to enable seamless internetworking not requiring complex gateways between the Internet and constrained networks such as WSNs, it is necessary to bring the Internet network layer protocols to these novel networks. However, WSN specific features such as limited battery power and memory size make the direct implementation of network layer protocols for sensor nodes impossible. For this reason, several organizations, including the IETF, have embarked on projects to design network layer protocols that enable efficient operation of WSNs and straightforward internetworking between these and the broader Internet. As far as the IETF is concerned, the two major standard efforts are the 6LoWPAN WG with its IPv6-like protocol, and the RoLL WG with its RPL routing protocol, both depicted in Figure 2.1. In this section, we briefly introduce 6LoWPAN and then focus our attention on the RPL routing protocol.

2.2.1 IPv6 in Low-Power Wireless Personal Area Networks

After approximately a decade of very active research in WSNs, the IETF chartered the 6LoWPAN working group to develop an IPv6-like protocol for these constrained networks using IEEE 802.15.4 at the physical and MAC layers. In order to be IPv6-compatible and work on top of IEEE 802.15.4, 6LoWPAN has to implement fragmentation, since IEEE 802.15.4 PHY frames have a maximum payload of 127 bytes, whereas IPv6 requires a 1280-byte minimum MTU. The standard implements fragmentation using a 3-field fragmentation header. Besides a tag field for keeping track of the IPv6 packet the fragment belongs to and an offset field to keep track of its position within the IPv6 packet, 6LoWPAN also tracks the datagram size with an additional fragmentation header field as this is useful to pre-allocate a buffer of the appropriate size on resource-constrained nodes [24].

Since IEEE 802.15.4 PHY frame payload is only 127 bytes long and MAC headers use up several of them, as many as possible of the approximately 80 remaining bytes should be dedicated to carry the IPv6 payload, not the header fields. For this reason, 6LoWPAN performs stateless header compression. The adopted solution is stateless to minimize complexity on resource-constrained nodes, and is based on assigning short representations for common values in header fields while removing redundant information at the link, network, and transport layers [24]. 6LoWPAN also uses assumptions on the link layer, such as that IPv6 addresses are derived from MAC layer ones, to implement the IPv6 neighbor discovery protocol [24] for WSNs. Thanks to its support of fragmentation, header compression, and simplified neighbor discovery mechanism, 6LoWPAN is a viable solution for IPv6-based networking in WSNs. As 6LoWPAN moved through the standardization process, the need for an effort to standardize a protocol to route 6LoWPAN packets became more relevant, and the IETF RoLL working group was chartered.

2.2.2 The Routing Protocol for Low Power and Lossy Networks (RPL)

RPL is the routing protocol for low power and lossy networks under development within the IETF RoLL WG. The working group defined four different application domains for this distance vector protocol: urban environments, industrial networks, home automation, and building automation [24]. As a consequence of the selected application areas, the protocol is optimized for convergecast, supports multicast, and makes unicast communications also possible [25]. In the current version of the protocol, there is no direct support for mobility [25]. Similar to what happened within the 6LoWPAN working group, the IETF RoLL WG had to make decisions as to what would be the most important use cases and scenarios. The decision to primarily target convergecast and not to support directly mobility are rooted in the analysis of the application scenarios and the need to limit the complexity and footprint of the protocol, so that it can be adopted in novel products and applications. The correctness of these design decisions is being validated during the standardization process, and will be put to the ultimate test when the standard is released and made available to be used in real-life applications.

2.2.2.1 RPL Basics

The protocol relies on an iterative process inspired by the Trickle algorithm [24] featuring one-hop DODAG (destination-oriented directed acyclic graph) Information Objects (DIOs) [25] used to propagate routing state. Instead of relying on a single node to relay packets to the root, sensors feature a parent set to achieve resilience to dynamically changing wireless links [24]. The actual next-hop neighbor is selected based on the metrics in the objective function used in the current instance of RPL [25]. RPL supports dynamic link metrics (for quality, latency, and throughput among

others) in DIO messages such as ETX (estimated number of transmissions for one-hop packet transfer) [26] [24]. Figure 2.5 portrays a DAG constructed by RPL. The root also uses DODAG Confirmation Objects (DCOs) to distribute root-defined network-wide parameters [25], which are used for instance in the mechanism to repair loops. Finally, optional security mechanism is proposed [24].

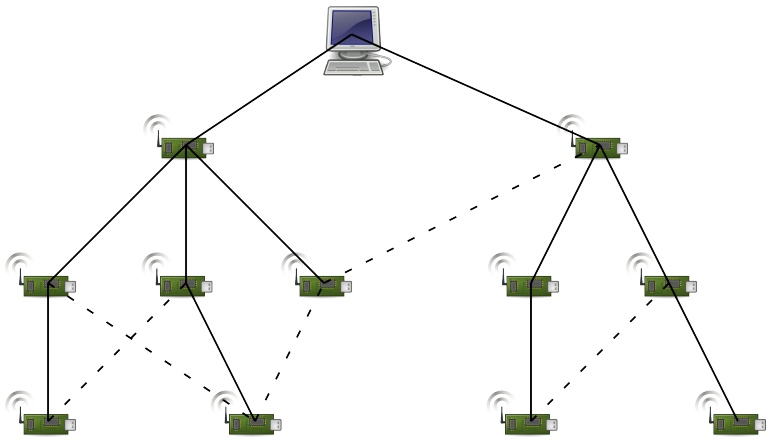


Figure 2.5. The directed acyclic graph constructed by RPL. Solid lines indicate the currently selected next-hop on the route to the root, while the dashed lines show the other nodes in the parent set.

2.2.2.2 Multicast and Unicast Communications

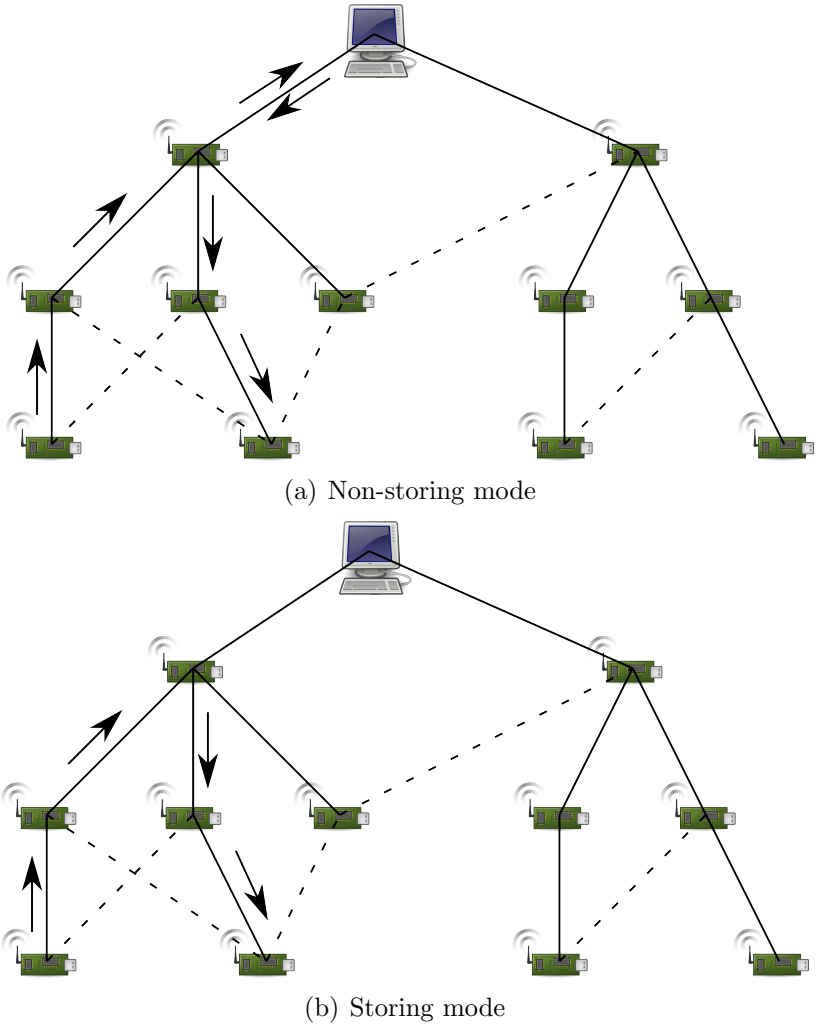
For multicast and unicast communications, RPL offers storing mode and non-storing mode [25]. In storing mode, nodes keep track of the forwarding nodes to all their descendants, so that they can re-route packets addressed to one of them, thus lowering congestion near the root. However, storing mode incurs into a larger memory footprint as nodes in the WSN must store the set of all their descendants as well as the corresponding forwarding nodes. In non-storing mode, all packets have to travel all the way to the root, which then source routes them to the destination. The

routes resulting from the two modes are depicted in Figure 2.6. The advantage of non-storing mode is that nodes are relieved of the need to store information regarding their descendants. Nevertheless, non-storing mode has to surrender in terms of bandwidth and route length what it gains in terms of memory footprint. In fact, in a source routing solution, information about all hops should be included in the packet header. Since the IEEE 802.15.4 MAC frame allows approximately 80 bytes for its payload and 6LoWPAN uses several of them for its other header fields, source routing can be implemented over approximately 8 hops if uncompressed IPv6 addresses are used, and still takes away valuable payload space for upper layer (*i.e.,* UDP) datagrams. For these reasons, we expect RPL instances to rely on storing mode more and more as memory size on resource-constrained devices slowly increases. Mixed operation with a subset of nodes using storing mode and the rest using non-storing mode is not supported [24].

2.2.3 RPL Implementations

In [27], the TinyRPL implementation of RPL and Collection Tree Protocol (CTP) [28] are evaluated by means of experiments on a testbed of 51 TelosB motes using BLIP, the Berkeley Low-Power IP stack, as the IPv6 implementation, where packets are generated every 5 and 10 seconds. Packet reception ratio is above 99.8% for both RPL and CTP, and between 8-10 control packets per hour are generated by each mote on average. The results also show that only 1.13 transmissions per hop and 1.86 end-to-end transmissions are required by TinyRPL. Unfortunately, the relevance of these results is hampered by the fact that no information is provided about the network topology, such as the average hop length of routes, on which they were collected. The authors also test the performance of bi-directional links set up by RPL, and show that a PC on the Internet sending requests to RPL motes through

the edge router receives a response approximately 98% of the times. These results on round-trip packet reception ratio seem to disprove the claim in [25] discussed previously that the RPL mechanism for route construction does not select reliable bi-directional links. Finally, the authors make several suggestions for the improvement of the standard and its implementation, including a stricter definition for Trickle



(a) Non-storing mode

(b) Storing mode

Figure 2.6. Unicast routing with RPL. In (a) non-storing mode, the packet is routed all the way to the root, whereas in (b) storing mode it is routed only up to the first common ancestor.

timer reset, trade-off solutions between storing and non-storing modes, and IPv6 fragmentation.

In [29], the authors present ContikiRPL, an implementation of RPL for Contiki OS, and discuss the results of simulations and experiments. According to the results, the ContikiRPL implementation uses approximately 3 KB ROM and 800 B RAM, which is more than an order of magnitude smaller than the 50 KB reported for it in [25]. The results for energy efficiency are encouraging as all motes maintain a duty cycle below 3% while generating 40 UDP packets per minute. However, similar to other experimental setups, the network size is limited to a few dozens motes and no information about the route hop length is provided.

2.2.4 RPL Analyzes

In [30], the authors survey the research work in the area of routing as it evolved from mobile ad-hoc networks (MANETs) to WSNs. In particular, they detail flooding protocols, clustering protocols, and geographical protocols, and then the so-called self-organizing coordinate protocols. For each protocol class, the authors provide a brief overview and describe its most relevant instances. They conclude their survey by presenting RPL, an instance of gradient-based routing in the class of self-organizing coordinate systems. Thanks to its chronological approach and classification of over 40 protocols in 4 well-defined classes, this survey offers a very good insight in the research work in the area of routing for WSNs.

A detailed survey of RPL is performed in [31]. After describing the features provided by RPL and the assumed network model, the authors present the mechanisms and messages used to build the routes from the root to the sensors (used for multicast), and from the sensors to the root (used for convergecast). Unicast between sensors is implemented by using these two sets of routes in what is called

dog-leg routing. The authors then present RPL mechanisms for route and loop repair, discuss several objective functions (performance metrics) used to provide QoS, and summarize security support in RPL.

RPL is experimentally evaluated on a small TelosB testbed running ContikiRPL with the Minimum Rank with Hysteresis Objective Function (MRHOF) and ETX. The results show that DODAG construction may take several minutes in a WSN of 30 nodes between 1 and 4 hops apart. The authors also measure an average power consumption of 2.2 mW during the construction of the DODAG in such a network. A packet loss ratio of 20% is observed when the RPL routes are used for multi-hop communications. The authors argue that other metrics may yield a better performance than ETX. Overall, they are satisfied with packet delays of 2.5 s in the 4-hop network. Finally, the performance of the reactive mechanism used for fault detection is also tested.

After reporting on these experiments, the authors describe other existing routing protocols and compare them to RPL. They also survey simulation and experiment results obtained by other researchers using several implementations, including ContikiRPL and TinyRPL. Finally, they point out some of the open issues in RPL, including the definition of appropriate objective functions and security mechanisms.

In [24], an overview of 6LoWPAN and RPL is presented. The survey first recaps how the research community did not consider the Internet architecture as a viable solution for communication in WSNs, thus developing many interesting, but also usually non-interoperable, ideas. It is argued that the push for the implementation of smart grids and home area networks for which WSNs are a core component prompted the adoption of the Internet architecture in WSNs. After this shift in opinion within the research community, the IETF chartered two working groups to define standards for IPv6 (6LoWPAN) and routing (RoLL) in these low-power and lossy networks, whose

efforts and proposed standards are then described. The survey also briefly describes BLIP 2.0 and TinyRPL, resp. the 6LoWPAN and RPL implementations for TinyOS. According to the authors, TinyRPL with non-storing mode (the implementation with highest memory requirements) uses approximately 9 KB ROM and 300 B RAM, thus being much smaller than ContikiRPL, which uses approximately 50 KB of memory according to [25].

In [25], a critique of the current version of RPL is offered. The protocol is first described, and then analyzed by the researchers who eventually support their statements with simulation and/or experiment results, whenever feasible. It is pointed out that traffic patterns other than convergecast are also common in certain application scenarios such as building automation, but RPL has limited support for them. Furthermore, complex metrics may bring about IP fragmentation as the ICMPv6 packets carrying RPL control messages may be larger than the approximately 80 bytes allowed by IPv6 on IEEE 802.15.4. Data traffic routed using RPL in non-storing mode may also risk fragmentation, as the route needs to be incorporated in the message. With respect to storing and non-storing modes to support downward routes, it is observed that storing mode limits the route length to 64 hops if IP fragmentation is to be avoided, while the non-storing mode restricts the network size to a few dozen devices as the ones near the root need to store paths to a large subset of the WSN in their limited memory. A proposed solution to this issue when operating in storing mode consists in assigning IP addresses in the sub-tree in a hierarchical fashion as it is the case in the Internet. However, this solution limits the ability of RPL routers to change preferred parent, as all the neighbors featuring in the parent set should share a common parent for the IP address hierarchy to be maintained.

As far as bidirectional links are concerned, it is argued that selecting a preferred parent based on the link quality from it to the RPL router may not be the optimal

solution as the quality of the link in the opposite direction may be very different. Furthermore, the Neighbor Unreachability Detection (NUD) mechanism proposed with RPL may be unable to detect whether the problem is indeed at this link and not farther away along the route, and to do so in a timely manner. The authors also criticize the complexity of RPL, and claim that most implementations will not be interoperable as they will have to pick a feature subset (as it is already the case for ContikiRPL [29]) in order to limit the memory footprint. They also criticize an insufficiently detailed specification, such as in the case of DAO message timing, which may lead to poor performance, and warn against Trickle performance in real-life WSNs, as its convergence is not as fast as stated by simulation results. While conceding that the RPL mechanism to support convergecast is elegant and well-understood, [25] also points out that reactive loop repair in RPL brings about potentially unacceptable delays and eventually packet losses if not all messages can be buffered at the RPL router while the loop is repaired. Furthermore, [25] also argues that mechanisms to enable unicast communication are underspecified and are likely to bring about IP fragmentation or require lots of memory for storing routes.

2.3 Application Layer

The World Wide Web is arguably one of the most successful applications enabled by the Internet. Among the several technologies making up what we call the Web, there are three fundamental ones: the HyperText Markup Language (HTML), the Hypertext Transfer Protocol (HTTP), and uniform resource identifiers (URIs). As discussed in [32, 33], HTTP implements the so-called representational state transfer (ReST) architecture thanks to which resources (often consisting of HTML-formatted data) are accessed via their URIs. In particular, ReSTful HTTP enables interaction with remote resources identified by their URIs via four basic methods: PUT, GET,

POST, and DELETE, used to create, retrieve, update, and remove resources, respectively. As the Internet of Things is slowly coming into being, researchers have started to design and analyze mechanisms to bring the powerful ReSTful paradigm to this new Internet that could potentially connect billions of devices across the world.

2.3.1 The Constrained Application Protocol (CoAP)

The Constrained Application Protocol is an application layer protocol that brings the ReST programming model of the Web to the Internet of Things and its embedded devices. Similar to HTTP, CoAP implements the four request methods of ReST; and uses similar response codes. By implementing the same ReSTful architecture as HTTP, internetworking between Web clients and CoAP-enabled WSNs as depicted in Figure 2.7 will be streamlined.

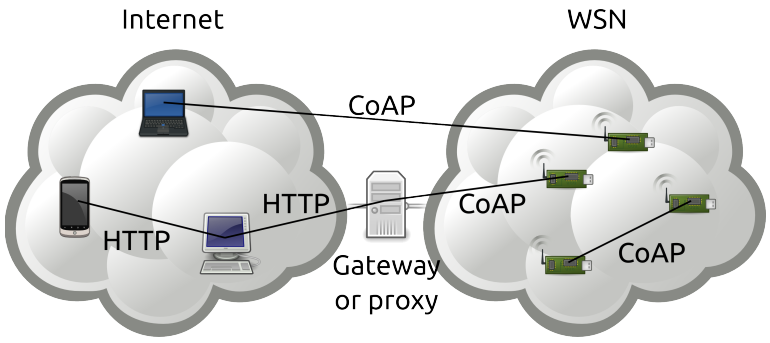


Figure 2.7. Internetworking via CoAP and HTTP between the Internet and WSNs.

As detailed in Figure 2.8, this will be made possible by using a simple gateway or proxy. In order to reduce complexity, CoAP relies on UDP instead of TCP, and defines its own simple mechanism to manage packet losses and retransmissions.

Figure 2.9 portrays the sequence diagrams for communication modes provided by CoAP. CoAP supports the transfer of large payloads such as it is the case when

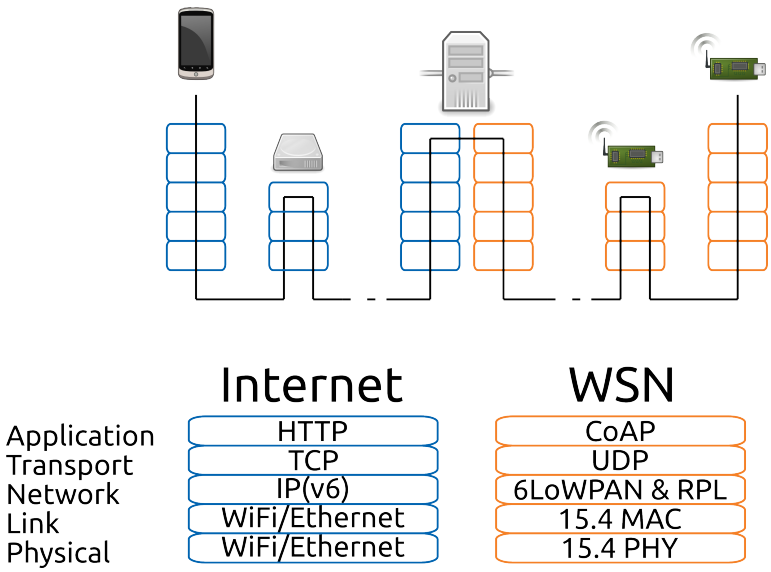
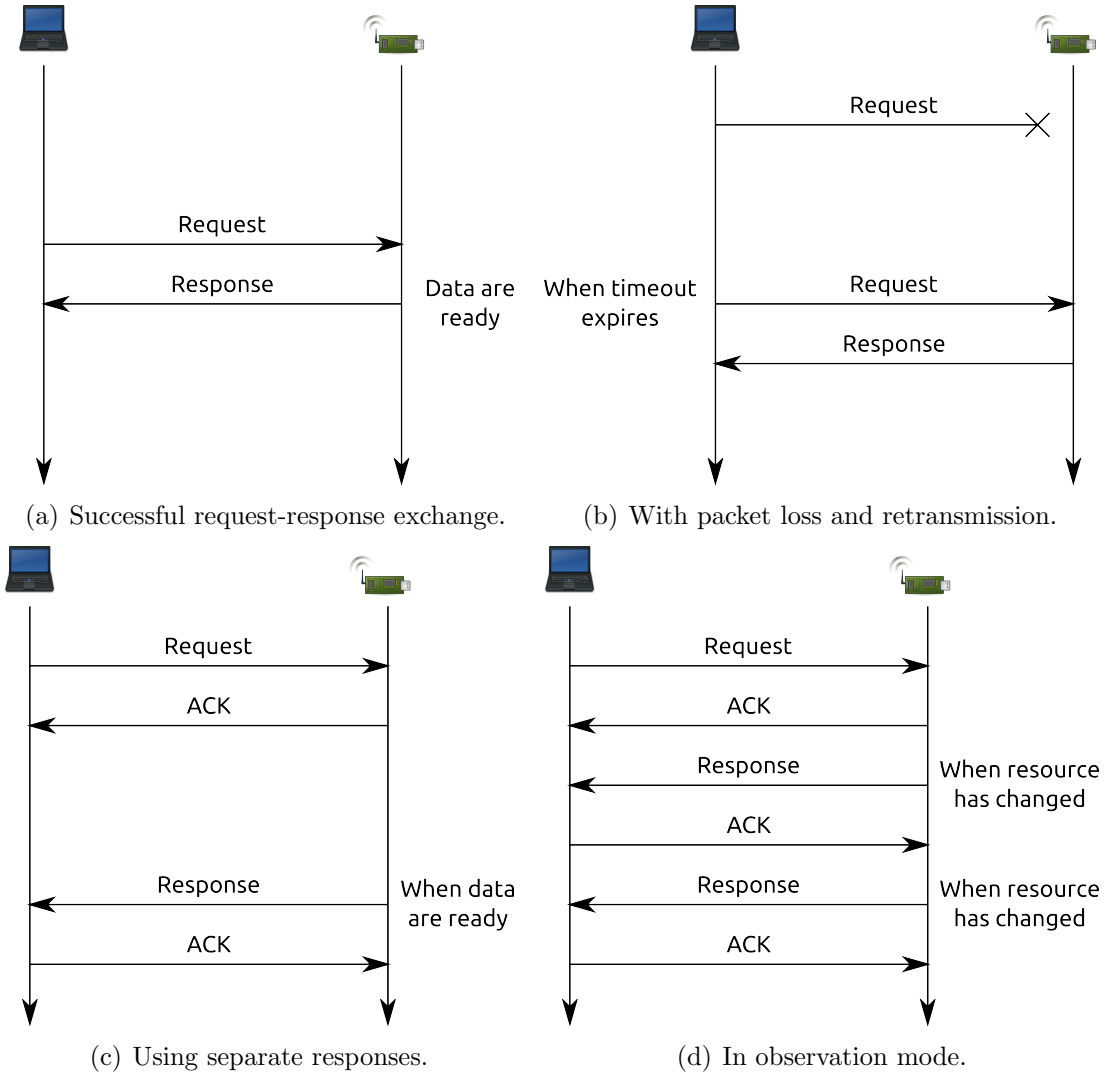


Figure 2.8. ReSTful networking between Internet device and sensor node.

the application or the firmware on the embedded devices need to be updated. Large payload transfer is achieved in CoAP by having multiple request-response exchanges in the so-called block mode, thus avoiding solutions involving IP fragmentation under UDP (although this is implemented by 6LoWPAN as described in Section 2.2), or having a stateful CoAP server. CoAP also provides a push-based mechanism, called observation, for monitoring a resource accessed via a GET request. In its GET request, the client asks the server to send a response with the current version of a resource not just once, but rather each time it changes. Finally, CoAP also addresses problems related to resource discovery for machines by defining standard resource paths on constrained devices.

In [34], the authors provide an introduction to the CoAP protocol. They first point out how standardization efforts at the network layer have brought IPv6 to WSNs (IETF 6LoWPAN), and are defining a common flexible routing protocol for these networks (IETF RoLL). Then they argue that an application layer protocol is needed that can support the growth of applications in the Internet of Things like



(a) Successful request-response exchange.

(b) With packet loss and retransmission.

(c) Using separate responses.

(d) In observation mode.

Figure 2.9. Examples of exchanges between CoAP client and server when (a) there are no errors, (b) the packet is lost, (c) the response is delayed, and (d) observation mode is active.

HTTP has supported the growth of the Web. After summarizing the features of ReST, the programming model underlying the Web, the authors introduce CoAP, discussing internetworking with HTTP, and block transfer, resource observation, resource discovery, and security in the protocol. An earlier introduction to CoAP by one of the authors of [34] is provided in [35].

In [36], the authors initially summarize the major features and issues in the Internet of Things that led to the design of CoAP, and offer a brief overview thereof. The rest of the survey presents and discusses the state-of-the-art of research on CoAP in several areas: performance evaluation, comparison between CoAP-over-HTTP and SOAP-based CoAP, tools and frameworks to ease development and usage of CoAP, solutions for network configuration and service discovery, and applications to building management and the smart grid. After listing CoAP-related applications and libraries, the authors point out that support for quality of service is missing in CoAP, and argue that the dominant design for the Internet of Things (*i.e.,* what combination of CoAP, SOAP, JSON, EXI, etc.) has not arisen yet.

2.3.2 CoAP Implementations

In [37], an implementation of CoAP for ContikiOS is presented. By relying on ContikiMAC [38], a sampling-based MAC layer protocol, it is possible to deploy an energy-efficient CoAP-enabled WSN. After summarizing MAC protocols with duty cycling and CoAP, the authors report on their implementation of CoAP for ContikiOS. This implementation provides all the protocol's major features, including block-wise transfer, resource observation, resource discovery, and separate response mechanism.

The authors run experiments on a linear 4-hop TmoteSky WSN with IEEE 802.15.4 and 6LoWPAN. The experiments show that energy-efficient operation of the CoAP-enabled WSN can be achieved simply by using an energy efficient protocol, in this case ContikiMAC, at the MAC layer without any changes to the application layer. In fact, the CoAP-enabled WSN can operate at a duty cycle around 1%, thus saving precious battery power, while latency is only lightly affected. However, the results also show that the rate of increase in latency for a CoAP exchange is higher

when a duty cycling MAC protocol is used. This implies that simply relying on a duty cycling MAC for energy efficiency may result in very high latency, if the route consists of more than just a few hops as in this experimental setup.

The authors also demonstrate how ContikiMAC can help limit latency in case of block-wise transfer or 6LoWPAN fragmentation for large CoAP payloads. In ContikiMAC, a sensor achieves this by signaling its next-hop neighbor that it will be sending a link-layer burst, *i.e.,* a series of frames, so that the neighbor stays awake and is ready to receive them right away without going through the channel sampling stage again.

In [39], an implementation of CoAP for TinyOS and Contiki is presented, along with its application to monitor a container and its content. After introducing the application scenario and the major features of CoAP, the authors describe libcoap, their C implementation of the protocol. As such it can be readily used for the communication between the WSN and the backend, thus reducing the amount of data to be transferred over a satellite or cellular link. In order to use the CoAP library on more constrained embedded devices, it had to be stripped of some features when it was being ported to ContikiOS and TinyOS. Unfortunately, the authors do not evaluate the performance of the proposed CoAP library over a multi-hop network, but just test it over a two-node TelosB WSN. Instead, to evaluate the proposed CoAP library, the authors compare the latency and amount of data transferred over a GPRS network, which has a round-trip time similar to that of a WSN, when using CoAP and different HTTP settings, including one with bare HTTP server and client over UDP. The results show that CoAP requires 107 bytes and 1.029 seconds, while bare HTTP over TCP uses 885 bytes and 3.076 seconds.

Another comparison of CoAP and HTTP is presented in [40]. The authors first recap the adoption and adaptation of IPv6 as the standard network protocol

in WSNs, and the ReSTful programming model. They then proceed to introduce two alternative stacks on top of 6LoWPAN, the IPv6 standard for embedded devices. One stack features TCP and HTTP similar to what is found in the Internet, while the other one uses UDP and CoAP. The two stacks are implemented in ContikiOS, and the authors use libcoap [39] and cURL (`http://curl.haxx.se`) as the respective clients to access resources on motes. In these experiments, the server and client are only one hop away from each other. The results show that CoAP exchanges consist of approximately between 10-20% as many bytes as HTTP ones, which is consistent with results presented in [39]. Furthermore, the authors also perform experiments using the Cooja simulator for Contiki to measure the energy consumption associated with the two stacks. The greater amount of bytes exchanged when using HTTP turns into a greater energy consumption for this protocol over CoAP. Preliminary results for transferred bytes and energy consumption were presented in [41]. The authors also perform simulations for varying request inter-arrival time and find that the energy consumption when using CoAP is not affected when requests become more frequent as it is the case for HTTP. However, the discussion of these specific results does not seem to be convincing. Finally, experiments on one-hop and two-hop routes confirm that CoAP achieves much shorter latency than HTTP.

CoAPP, an implementation of CoAP for TinyOS, is presented in [42]. Both server and client interfaces are provided in the CoAPP component with TinyOS commands in the client triggering TinyOS events in the server, and vice versa. Experiments show that 20 servers on a MEMSIC TelosB can successfully serve 90% of the 50 requests per second sent by another TelosB. Therefore, the proposed CoAP server and client implementation is deemed to be effective and scale well. The authors also implemented a library to support the encoding and decoding of XML documents into and from EXI data streams. For the EXI processing library to be used

on resource-constrained microcontroller such as the Texas Instruments MSP430, the XML schemas need to be pre-processed into a set of grammars and data structures. Experiments show that the proposed EXI library is very efficient as the size of the output EXI data stream is usually around 10% of that of the original XML document containing sensor data, if a byte-aligned schema is used.

2.3.3 Internetworking Between HTTP and CoAP

A gateway and a proxy for ReSTful internetworking of CoAP and HTTP are introduced in [43]. A preliminary design of the gateway was presented in [41]. The proposed gateway consists of a Web server that presents a HTTP interface to the Web client and a CoAP client running ContikiOS that interacts with CoAP-enabled devices in the WSN. Web clients, such as browsers, are unaware of CoAP, and retrieve data from the WSN by connecting via HTTP to the Web server. The Web server then retrieves cached data from a database (Apache CouchDB [44] in this case), or uses the CoAP client to pull data from the deployed sensors.

Since a HTTP/CoAP gateway is a complex system, the authors also propose a simple HTTP-CoAP proxy. Given that both HTTP and CoAP implement the ReST programming model, the development of a HTTP-CoAP proxy is relatively straightforward. In fact, the proposed proxy offers a fully transparent protocol-agnostic resource access, so that any Web client can access a WSN using HTTP using this proxy. While the authors state that the proposed proxy does not implement resource observation, they do not list which other features have been implemented.

2.4 Discussion

Although research work on the MAC, network, and application layers of the wireless communication stack for WSN has brought about several important results,

there are still several open problems in this area of research. In the rest of this section, we break down the discussion of the communication stack along the each layer and point out several open issues and problems for each one of the surveyed layers.

2.4.1 MAC Layer

At the MAC, many protocols for each one of the traffic classes have been proposed. As far as the MAC layer is concerned, an interesting question is whether the flexibility offered by IEEE 802.15.4 is sufficient for the diverse WSN applications looming ahead. In particular, it will be important to establish whether this standard MAC protocol can meet the requirements in terms of network lifetime for environmental monitoring applications, as well as the requirements in terms of delay and throughput of industrial applications. If research prototypes, but even more so, commercial applications demonstrate the effectiveness of IEEE 802.15.4 to achieve these different and conflicting objectives, then the standard will be widely adopted. Otherwise, there is the risk of a fragmentation in terms of the adopted MAC protocols. In that case, it may be beneficial to revisit the standard and define a more flexible solution such that nodes in a WSN, or a subnetwork thereof, can switch to the optimal operating mode within the standard for the current traffic load, be it heavy, medium, or light. Given the diversity of real-life application scenarios, we would not be surprised if they required performance levels beyond those achievable with IEEE 802.15.4.

Independent of the potential need for a more flexible standard MAC protocol, there is another important question regarding MAC protocols that has not been fully answered yet. Although MAC protocol classes have been defined for different traffic loads, there is no algorithm that can select the optimal MAC protocol given the observed traffic in a WSN. Such an algorithm should take into consideration the

characteristics of the traffic. The algorithm could be employed not only for network-wide pre-deployment MAC protocol selection, but also for dynamic changes both in space and time. For instance, a preamble sampling-based protocol could be initially selected for the WSN, while a subnetwork may switch to a scheduled protocol later on to carry heavy traffic loads in a more energy-efficient manner than the original protocol. This behavior is portrayed in Figure 2.10. Although there exist hybrid protocols and solutions that offer a simple version of this (*e.g.,* Funneling MAC [23] with TDMA in subnetwork near the sink and CSMA in rest of the WSN), they address special situations, and do not provide a mathematical proof of their performance. This is undoubtedly a very complex problem, but advances in this area are likely to greatly benefit WSN applications by improving most performance metrics, including delay, throughput, and lifetime.

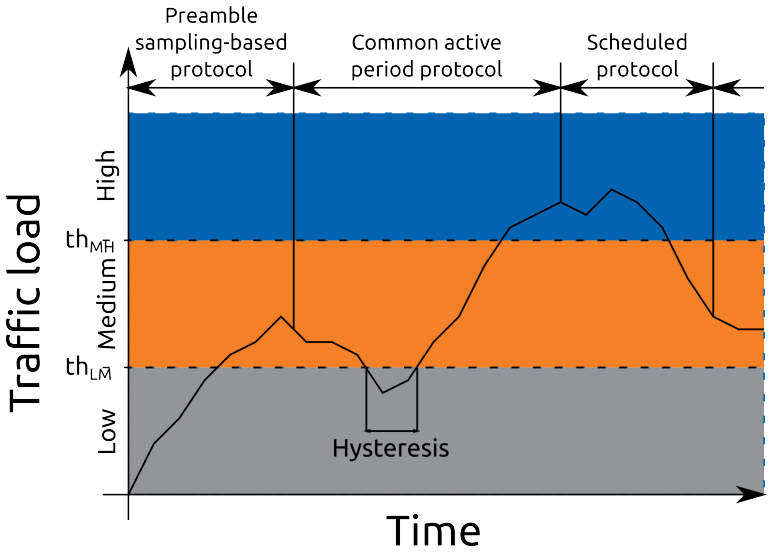


Figure 2.10. Example of dynamic MAC protocol selection for time-varying traffic load.

Finally, IEEE 802.15.4 or any other proposed standard MAC protocol should be extensively analyzed within the broader horizon of the communication stack. To provide the necessary background, in the following sections we introduce and discuss recent developments and advances at the network and application layers to help with this task.

2.4.2 Network Layer

After more than a decade of research work in the very important area of routing for WSNs, we are finally witnessing a strong effort by the IETF to design a standard protocol. The slightly earlier definition of an IPv6-like protocol for the WSN network layer, namely 6LoWPAN, gave the necessary boost for the chartering of IETF RoLL. Unlike application-specific protocols such as the ones in the ZigBee or Z-Wave stacks, RPL is designed to be deployed in several different scenarios, from home automation to industrial environments. Given the important role of the network layer in the Internet communication stack, it is instrumental for the growth of WSNs that the protocol be not only appropriately timed, but also flexible enough to be used in more than just one vertical market. For this reason, it is instrumental that open issues and problems be addressed swiftly during the standardization process.

An important issue is the support of traffic beyond convergecast, in particular unicast to nodes in the WSN. As we describe in more detail in Section 2.3, the Internet of Things is thought of as an extension of the current Internet where data collected from embedded devices such as sensor nodes can be remotely accessed via CoAP, an application layer protocol similar to HTTP. In this scenario, there is a need for requests to be sent through the WSN root over a multi-hop route to individual sensor nodes. However, the current support for this kind of communications in WSN relies on the storing or non-storing modes described above. Therefore, we argue that the

performance of RPL in storing and non-storing modes be more thoroughly evaluated using CoAP traffic. Due to their shortcomings, it may be that these modes do not enable proper functioning of CoAP. In this case, the protocol should be promptly revised, and other solutions to better support unicast, such as the one suggested in [25], should be considered.

Overall, there is a need for more implementations of RPL beyond the two for TinyOS and ContikiOS, and extensive experiments on larger testbeds. In fact, several issues may arise when the protocol is tested on real-life larger deployments. First of all, there may be problems with the stability of the routes. Since the wireless medium is very noisy and its dynamics highly variable, it is unknown whether the RPL mechanism to select a forwarding node within the parent set is both optimal and stable enough in all the different application domains, from suburban homes to factory floors. In fact, it is likely that the current metrics (*e.g.,* ETX) and objective functions (*MRHOF*) bring about frequent route updates in real-life noisy environments. As a result, timely communications in WSNs consisting of a few dozens nodes and several hops between root and leaf may become impossible. Once the performance of the protocol and, more specifically, the link metrics and objective functions is assessed, novel objective functions may be needed, that achieve (sub-)optimal but stable routes.

2.4.3 Application Layer

Although the CoAP protocol is still in the stage of Internet Draft and a standard has not been proposed yet within the IETF CoRE WG, the community around it is overall very active. This activity is demonstrated by the surveyed articles, including several ones presenting implementations of the protocol. The many analyzes and implementations of the protocol also bring out several directions of further research.

First of all, we argue that the existing body of work on experimental analysis of the protocol falls short of thoroughly validating CoAP, especially when other communication stack layer are considered. For instance, while it is reassuring to know that CoAP (which uses UDP) is indeed better than HTTP (which uses TCP) in terms of transferred bytes, energy consumption, and latency, [40] this was somewhat expected as it was one of CoAP's goals from its inception. Since implementations of CoAP are already available for two of the major WSN operating systems, namely TinyOS and ContikiOS, it would be beneficial to perform an extensive experimental evaluation of this protocol. In terms of comparison of different implementations, [39] is a welcome first step, but additional effort should be dedicated to this task. Most importantly, CoAP should be evaluated on larger testbeds. Small setups such as the one consisting of 4 nodes on a path used in [37] can provide an initial validation of the protocol, but results obtained on them cannot be taken as a final proof.

Another important open issue is the necessity of energy-efficient mechanisms at the application layer. As reported in our survey of existing experiment results, it appears that the usage of an energy-efficient MAC protocol such as ContikiMAC is sufficient to greatly improve the energy efficiency of the whole communication stack [37]. While this is an important finding, we argue that it is insufficient to discard the pursuit of energy-efficient solutions at the application layer. In fact, the results in [37] were obtained for a specific traffic load, MAC protocol, and network topology. However, as we remarked in Section 2.1, different combinations of traffic load and MAC protocols present greatly varying behavior. For this reason, we argue that more extensive experiments with CoAP on different network topologies, or at least all traffic classes should be performed. Only the experiment results will show if the behavior observed in [37] for ContikiMAC and CoAP in presence of a relatively low traffic load extends to heavier loads and different classes of MAC protocols. In case these

experiments highlight a significant performance degradation, countermeasures will have to be adopted. First of all, existing mechanisms within CoAP may be employed. For instance, separate responses could be used to counteract the increased number of retransmissions that would derive from timeouts at the client side. Alternatively, CoAP should be re-assessed and extended with novel mechanisms to support more energy-efficient operations.

Although the proposed and existing standards try to accommodate different use cases, not all application scenarios can be optimally addressed even by the most flexible standards. For instance, the proposed standards for the communication stack do not readily support in-network fusion, because the content of packets on their way from sensors to the base station cannot be inspected and modified, unless the boundaries between layers in the communication stack are broken. We argue that in most application scenarios the advantages of standardized solutions, such as interoperability of different systems, will be preferred over the positive features of customized solutions, such as a slightly reduced cost. Therefore, any solution involving in-network fusion should design, implement, and optimize it at the application layer while relying on the standard protocols at the underlying layers, rather than proposing customized cross-layer approaches.

2.5 Summary

In this chapter, we introduced several protocols and solutions developed to support communications in WSNs. We focused especially on the MAC, network, and application layers, due to their relevance within the communication stack. After pointing out a slow convergence of different solutions towards an Internet-like WSN communication stack featuring IEEE 802.15.4 at the physical and MAC layer, IETF 6LoWPAN and IETF RPL at the network layer, UDP at the transport layer,

and IETF CoAP at the application layer, we discussed specific protocols and solutions more in detail. We observed that the research community is very active in the synthesis of many research ideas, which were proposed in the past 15 years, into well-designed standard protocols. In our discussion, we pointed out several open problems, including the selection of optimal MAC protocol for a given traffic load, objective functions that select stable routes, and the importance of energy-efficient mechanisms at layers beyond the MAC one. All these problems require more experiments to be fully modeled, and novel ideas to be solved. To conclude, we argue that, now more than ever, novel ideas solving these open problems will have the opportunity to shape standard protocols and the WSN applications of the (near) future.

CHAPTER 3

A SEMI-DISTRIBUTED LOCALIZATION PROTOCOL FOR WIRELESS SENSOR AND ACTOR NETWORKS

Despite the tremendous potential for a multitude of application domains, wireless sensor networks pose unique challenges in the design of protocols for their operation. This is due to the network characteristics, such as the high-density deployment of sensors, anonymity of individual sensors, limited resources including battery energy budget per sensor, and possibly hostile environment. For example, the design of localization protocols is critical for the collected data because sensor nodes may not know their actual position. In fact, the sensed data might be meaningless unless related to the exact position or at least a sufficiently small region of the monitored area. Moreover, the limited energy budget requires the design of energy-efficient protocols to prolong the network lifetime, thus forcing the sensors to alternate between sleep and awake periods.

In addition, efficient techniques must be developed to allow sensors to communicate with the outside world, the receiver of the data harvested by the WSNs. A viable solution involves the use of one or several *actors*, *i.e.*, special long-range radios deployed along with the sensors. Each actor has a full range of computational capabilities, can send long-range directional broadcasts to the sensors up to distance R, can receive messages from nearby sensors, and has a steady power supply. This implies that the sensor network must be able to communicate with multi-hop paths and only a small number of sensors have an actor as one of their one-hop neighbors. The actors organize the sensors in their vicinity into a short-lived, *actor-centric* network

in support of a specific mission; when the mission terminates the network is dissolved and the sensors return to their unorganized state. As an example, imagine a blind person attempting to cross a street in a sensor-instrumented city block. The sensors in his/her immediate vicinity organize themselves into a short-lived network whose stated goal is to help the pedestrian to chart a safe course to his/her destination. Once the person has been assisted, the sensor network is disbanded [45]. Such WSNs, in which sensors and actors collaborate to accomplish a mission, are called *wireless sensor and actor networks* (WSANs) [46, 47].

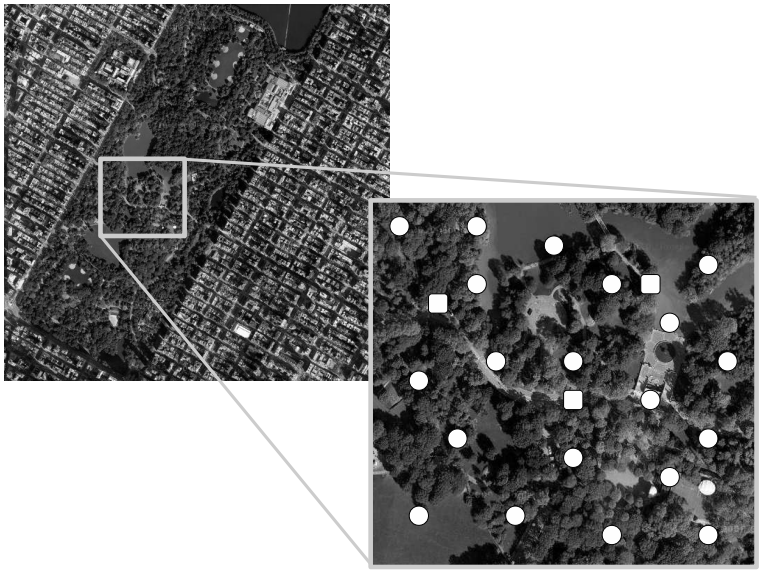


Figure 3.1. In environmental monitoring applications with mobile nodes, randomly deployed static wireless sensor nodes (white circles) collect data and relay them to mobile actors (white squares).

In this chapter, we investigate mission-oriented WSANs with an additional feature: the sensors follow a periodic working schedule. In such networks, called *duty-cycle wireless sensor and actor networks* (DC-WSANs), the actor calls for a specific mission, while the sensors in the actor's vicinity asynchronously wake up and

follow a working schedule for the rest of the mission duration[1]. In other words, the sensors alternate between awake and sleep periods of fixed lengths. During the awake period, the sensor is active and hence can sense, transmit/receive data to/from other sensors or the actor. During the sleep period, all the sensor components are inactive, except for the internal clock.

Our work is mainly addressed to environmental monitoring applications (see Figure 3.1), where the sensed data are collected by wireless sensors densely deployed over wide areas that can span several square kilometers in urban, rural, or natural environments. In such applications, sensors are usually deployed by an unmanned vehicle and remain unattended in a vast, possibly hostile, geographical area for long periods of time. The sensors are periodically awake in order to sense the area to be monitored. According to a predefined monitoring schedule, actors can roam the environment to collect data because effective data communications over the entire network are difficult to achieve due to the large number of communication hops [50, 2]. Abstracting from this real world application, we envision a DC-WSAN as shown in Figure 3.2, where each actor is mobile and, upon reaching a specific location, stations there to organize an actor-centric network (gray disks in Figure 3.2) for collecting sensed data.

This chapter deals with one of the fundamental problems arising in the context of DC-WSAN: localization. Our solution methodology relies on network modeling at different levels of abstraction. In particular, we model the DC-WSAN as a dense network whose underlying graph has a vertex for each sensor and an edge for each pair of sensors that can directly communicate. Since the sensors follow a working

[1]The research presented in this chapter has been published in the Proc. of the 8th IEEE International Conference on Pervasive Computing and Communications (PerCom 2010) Workshops [48], and Wiley Networks [49].

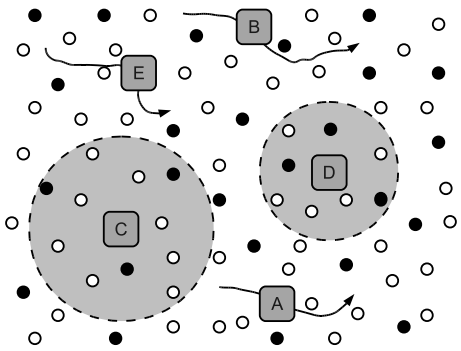


Figure 3.2. A DC-WSAN featuring several awake sensors (black circles), sleeping sensors (white circles), and a few actors (squares). Actors C and D are temporarily static and have corresponding actor-centric subnetworks (gray disks), while actors A, B, and E are moving to new positions as indicated by their trajectories.

schedule, such a network is dynamic, that is, an edge appears only when the involved sensors are simultaneously awake.

3.1 Related Work

Wireless sensor networks have been extensively studied in recent years. By reducing the deployment overhead in terms of unit cost and installation time, the development of dense distributed sensor systems has become feasible and dense networks can be active all the time even though individual sensors may follow a working schedule alternating between sleep and awake states. In this way, the network noticeably extends its lifetime while sensors save energy.

The objective of localization is to provide sensors with geographic coordinates or with a position relative to a node in the network itself [2]. A straightforward but costly solution may use global positioning system (GPS). Other solutions may assume the existence of anchor nodes which are aware of their location and allow other sensors to infer their locations by exchanging information with them.

Although the above-mentioned solutions attempt to provide the exact position for each sensor, in several applications sensors only require *coarse* positioning

with respect to a reference point. Location training is a family of techniques for coarse positioning. Recently, a number of papers presented location training protocols which achieve coarse-grained localization by imposing a polar coordinate system (see Figure 3.3) by an actor [51, 45, 52, 53]. Such a coordinate system divides the actor-region into h equiangular circular sectors and k concentric coronas (*i.e.*, areas between two concentric circles both centered at the actor) with equal widths, and the coarse-grained position of a sensor is given by the corona and the sector where it resides. Clearly, many sensors share the same position. For example, in Figure 3.3, all the sensors in corona 3 and sector 0 learn position $(3, 0)$. In all algorithms except the distributed one in [53], sensors compute coarse-grained positions based on the information received from the actor without performing any additional communication.

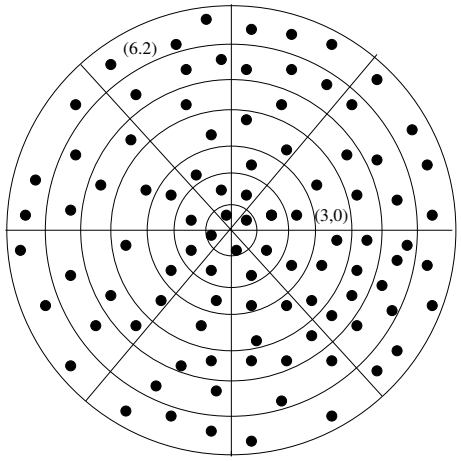


Figure 3.3. The ordinary virtual infrastructure with $k = 7$ coronas and $h = 8$ sectors.

In particular, the centralized protocol presented in [52] assumes that all the sensors are synchronized to the master clock running at the actor and that sensors are *aperiodic*, that is, their sleep-awake schedules depend on the computation required for the localization protocol. The *Flat* localization protocol and its variations, pre-

sented in [51], are suitable for sensors that harvest energy from the environment and thus follow *periodic* sleep-awake schedules. The protocols in [51, 52] apply to sensors which are *homogeneous* in terms of sleep-awake schedules. In contrast, the training protocols proposed in [45] can simultaneously handle both *aperiodic* and *periodic* sensors. Finally, a new distributed protocol for training, called *Cooperative*, is proposed in [53]. This is the first protocol that exploits the sensor co-operation in massively and randomly deployed sensor networks. The key idea is to train only a subset of sensors, uniformly distributed in the network. Then, by means of local inexpensive transmissions, the trained sensors relay the acquired coordinates to the largest number of remaining untrained sensors in their vicinity. Since each sensor stays awake for a *constant* number of awake periods, the *Cooperative* algorithm performs better than previous algorithms, and it is suitable for dynamic contexts where the actor moves and the actor-centric network needs to be quickly and often re-organized. In this work, a thorough analysis of the density required by a new simplified version of the co-operative process is conducted both theoretically and experimentally, thus demonstrating that the distributed protocol is indeed feasible and of practical interest.

3.1.1 Our Contributions

In this chapter, we assume that sensors wake up randomly for the first time and then maintain their own periodic sleep-awake schedule, independent of the network status or of the protocol they perform. Under such assumptions, we investigate one of the fundamental aspects of the actor-centric network, namely the organization of the sensors into a virtual infrastructure with clusters spanning over similar areas.

We adopt a new generalized polar coordinate system. The coordinate systems employed in training protocols so far have the same number of sectors in any corona, although the areas of the coronas constantly increase moving out from the center

to the fringe of the actor-region (see Figure 3.3). When the number of sectors is fixed, the clusters in the outer coronas have an area greater than those in the inner coronas. We instead propose to keep the areas of clusters almost equal to that of the inner-most corona. To impose such a new coordinate system (see Figure 3.4), we investigate a simplified version of the localization algorithm proposed in [53]. The protocol consists of two phases: in the first phase, a subset of sensors, referred to as *seeds*, are trained in a centralized manner by the actor; in the second phase, seeds broadcast the acquired location information to their neighbors for two periods. The second phase is fully distributed. In this chapter, we provide the mathematical analysis of the sensor density in each corona required to guarantee the effectiveness of the algorithm. We show that, when the sensors are deployed according to a *Poisson point process* [54], all of them are trained with high probability, if the sensors deployed in a region of area πr^2 follow a Poisson distribution of parameter $L \ln L$, where r is the sensor transmission radius for intra-cluster communications and L is the period of the sleep-awake schedules. Thus, the required average number of sensors in each neighborhood is only a function of the length L of the sleep-awake cycle of each sensor because we need to guarantee a local connectivity within a sensor's communication radius, and not the connectivity of the whole network. As a consequence, our analysis improves on the preliminary result derived in [53].

The remainder of this chapter is organized as follows. Section 3.2 defines the network model. Section 3.3 presents the localization protocol by specifying the actor and sensor behaviors. Section 4.3 presents a thorough analysis of the conditions under which the algorithm guarantees, with high probability, the propagation of the location information to all sensors. Simulation results that confirm the analytical results for localization are also presented in Section 3.3.3.

3.2 Models

Let us describe the virtual infrastructure to be imposed on the actor-region and the underlying model for both actor and sensor communications, sleep-awake schedules, and sensor deployment model. From now on, let $|i|_j$ denotes the *modulo* operation, that is the non-negative remainder of the division of i by j.

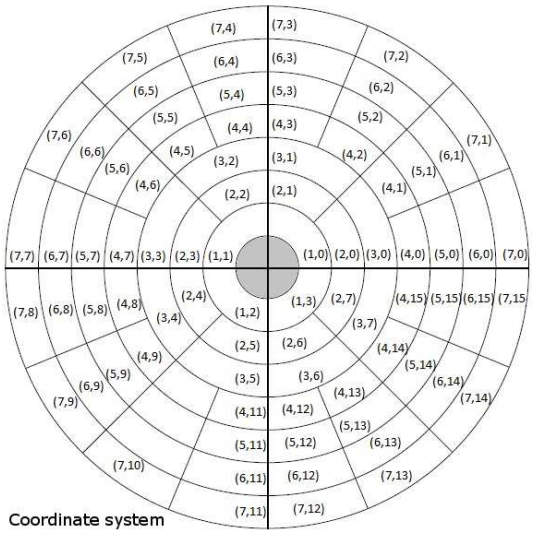


Figure 3.4. The virtual infrastructure with numbered clusters when $\ell = 4$.

3.2.1 Virtual Infrastructure

We define a new virtual infrastructure with clusters of roughly the same area. It consists of $k \geq 2$ *coronas*, numbered from 0 (the innermost) to $k-1$ (the outermost). Corona c is partitioned into h_c sectors, where

$$h_c = \begin{cases} 1 & \text{if} \quad c = 0 \\ \ell & \text{if} \quad c = 1 \\ \ell \cdot 2^{\lfloor \log_2 c \rfloor} & \text{if} \quad 2 \leq c \leq k-1 \end{cases} \tag{3.1}$$

Indeed corona 0 is not divided into sectors as it is managed by the actor, while corona 1 will be divided into a preset number of ℓ sectors. Then, the number of sectors will be doubled at each corona $c = 2^p$, $0 < p \leq \lfloor \log_2(k-1) \rfloor$ because corona $c = 2^p$ has area $\pi(2^{p+1}+1)$ which is almost the double of the area $\pi(2^p+1)$ of corona $c = 2^{p-1}$. This partition guarantees that the area spanned by a cluster in corona 1 is at most the double of any other cluster in corona $c > 1$. Figure 3.4 illustrates the virtual infrastructure when $\ell = 4$. The sectors in corona c are numbered from 0 to $h_c - 1$ in counterclockwise direction, starting to count from the sector above the horizontal x-axis.

Noting that the outermost corona $c = k-1$ will be divided into $h = \ell \cdot 2^{\lfloor \log_2(k-1) \rfloor}$ sectors, the virtual infrastructure can be viewed as an ordinary polar coordinate system with k coronas and h sectors, in which the inner coronas just ignore further subdivisions of their coronas.

3.2.2 Actor Communications

Each actor is equipped with two antennae: one isotropic and one directional. The transmitting power w^a_{Tx} of the *isotropic antenna* can be modulated in the interval $[w^a_{Tx,min}, w^a_{Tx,max}]$, for which we assume $w^a_{Tx,min} = 0$. This enables the actor to reach any circle c of radius $\rho(c) \in [0, R]$ centered at the actor's position, where R is the radius corresponding to the maximum transmitting power $w^a_{Tx,max}$. Supposing the existence of a bijective function $g(\cdot)$ that maps transmitting power and circle radius according to the radio propagation model, the circle radius $\rho(c)$ due to communication of transmission power w^a_{Tx} can be computed as $\rho(c) = g(w^a_{Tx})$. Similar to the transmitting power of the isotropic antenna, the transmitting angle ϕ^a_{Tx} of the *directional antenna* can be set in the interval $[\phi^a_{Tx,min}, \phi^a_{Tx,max}]$, for which we assume $\phi^a_{Tx,min} = 0$ and $\phi^a_{Tx,max} = 2\pi$. This enables the actor to reach circular sectors of different size

around its position. Given the above assumptions, we define the *actor region* as the complete disk of radius R covered when the actor transmits at the maximum power $w^a_{Tx,max}$. Finally, we assume that the actor can directly retrieve the messages that have reached corona 1.

3.2.3 Sensor Communications

Each sensor is equipped with an isotropic antenna for which only a pair of transmitting power settings $(w^s_{Tx,min}, w^s_{Tx,max})$ is available. Transmitting power $w^s_{Tx,min}$ enables a sensor to reach a disk of radius $r = g(w^s_{Tx,min})$ such that r is much smaller than the width $\frac{R}{k}$ of each corona of the polar coordinate system, and thus it is used for *intra-cluster* communications. The intra-cluster communications occur, for instance, during the distributed phase of the localization algorithm. The second transmitting power setting $w^s_{Tx,max}$ enables a sensor to cover the *corona width* (*i.e.*, radius $\frac{R}{k}$) thus making such a power suitable for *inter-cluster* communication. The inter-cluster communications occur, for instance, in the routing protocols or during the leader election algorithm. From now on, for the sake of simplicity, we assume that the corona width $\frac{R}{k} = 1$, and thus the radius of the actor region $R = k$.

As regard to the communication channels, sensors have the ability of transmitting and receiving over different frequency bands, or *channels*. *Frequency division multiple access* (FDMA) is assumed in our model to enable continuous transmission in the time domain, and it will be proved that no more than 2ℓ channels are needed in our protocols if ℓ is the number of sectors in corona 1 (see Eq. (3.1) in Section 3.2.1). Such a number of channels can be easily obtained in most existing sensor chips. Currently, typical radio communication chips support a number of channels that vary from 16 to 169 [55].

We assume that each sensor transmits always on the same channel, which depends on the cluster where it resides, but it listens to communications on its transmission channel, as well as on the channels of all adjacent clusters. Thus, sensors have one active queue for each channel they can use in reception. If sensors do not have the ability of transmitting and receiving over different frequency bands, they can work in *time division multiple access* (TDMA). Each time slot is expanded into several sub-slots, with one sub-slot for each different channel. A sensor transmits always on the sub-slot associated with its channel, but it can listen to its transmission sub-slot as well as to the sub-slots associated with the channels of all adjacent clusters.

A collision happens when two sensor transmissions using the same channel are simultaneously received by a single sensor. We assume that intra-cluster messages are concordant and are received correctly even if simultaneous transmissions collide at a receiving sensor, whereas inter-cluster messages, if discordant, collide and get corrupted. Therefore, to avoid message collisions, a channel assignment problem has to be solved whose aim is to avoid that two clusters, say A and B, adjacent to the same cluster, say C, are assigned to the same channel. In fact, if this is the case, simultaneous and discordant transmissions of the sensors in A and B will use the same channel and will collide at the receiving sensor in C.

We also assume that actor and sensors share at least one channel and that the actor messages are tagged with a specific marker so as to distinguish them from packets originated from sensors.

3.2.4 Working (or Sleep-Awake) Schedules

As far as working schedules are concerned, we assume that they are periodic. Each period lasts for L time slots, during which a sensor is awake for $d \geq 2$ time slots[2] and sleeps for the remaining $L - d$ time slots. Moreover, for the sake of simplicity, we assume that L divides k, that is $|k|_L = 0$.

Although sensors and actor use equally long, in-phase time slots, they do not necessarily wake up at the same time slot t. Denoting the 0-th time slot as the one when the actor wakes up, each sensor asynchronously wakes up for the first time in one time slot chosen uniformly at random during the interval $[0, \ldots, L-1]$. Formally:

Definition 1. *A sensor of* time zone z*, for* $0 \leq z < L$*, is a sensor that is awake in each time slot* t *such that* $|t|_L = z + i$*, with* $0 \leq i \leq d - 1$*, and sleeps in each time slot* t *with* $|t|_L = z + i$ *with* $d \leq i \leq L - 1$*. We use notation* $z(s)$ *to indicate the time zone of sensor* s*.*

In order to save energy, a sensor can skip the awake state staying in sleep mode for L time slots, if it is not required to be active in a period. Moreover, at any time slot t, only sensors belonging to the d time zones $|t - (d-1)|_L, \ldots, |t|_L$ are simultaneously awake.

During the leader election, a sensor is active only in the first time slot of its awake period. Moreover, when a sensor is selected as a leader, it dedicates the first two awake time slots of its sleep-awake period to the inter-cluster transmissions. Thus, a leader of time zone x listens to the incoming channels in its first awake time slot, *i.e.*, any time slot t such that $|t|_L = x$, and transmits in the second time slot, *i.e.*, $|t|_L = |x+1|_L$, using the channel assigned to the cluster where it resides. This implies

[2]The requirement of at least 2 time slots for awake time is explained in the analysis of the algorithm in Lemma 1.

that a message transmitted by a leader of time zone x is only received by a leader of time zone $|x+1|_L$.

3.2.5 Deployment Model

The sensor geographical positions can be modeled by a 2-dimensional *Poisson point process* [56]. Precisely, denoting the density of the Poisson point process as Λ, where Λ is measured as the number of sensors per unit area, the number $N(A)$ of sensors in a region A follows a *Poisson probability distribution* of parameter $\Lambda||A||$, where $||A||$ represents the area of the region A. Thus, the probability that ψ sensors reside in region A is given by:

$$P(N(A)=\psi)=\frac{e^{-\Lambda||A||}(\Lambda||A||)^{\psi}}{\psi!} \tag{3.2}$$

Furthermore, since sensors are uniformly distributed among L time zones, the number of sensors within the same time zone z in a region A can be modeled by a Poisson distribution of parameter $\frac{\Lambda||A||}{L}$, as shown in [54]. Therefore:

Definition 2. *In the actor region of area πR^2, we assume that there are on the average $N=\Lambda\pi R^2$ sensors, which are partitioned into L groups, one for each time zone, and each group has on average size $\frac{N}{L}=\frac{\Lambda\pi R^2}{L}$ where $0\leq z\leq L-1$. Thus, at each time slot, on average there are $d\frac{N}{L}$ awake sensors in the network.*

Clearly, the same stochastic argument can be adopted to count the average number of sensors in the area covered by an inter-cluster sensor transmission.

3.3 Localization Protocol

The coronas and sectors are imposed by using, respectively, the isotropic and the directional antennas. Since the procedure to impose h sectors is the same as that to impose h coronas once each isotropic broadcast of radius a is replaced by a

directional broadcast of angle $a\frac{2\pi}{h}$, let us focus on the corona localization protocol only. First, we call a sensor *covered* when it has acquired corona location information. If covered, a sensor is *trained* when the location is correct, otherwise it is *mistrained*. Moreover, by time zone we will also mean the sensors that belong to it.

In this section, we present the pseudo-code of the proposed localization algorithm. Then, in Section 4.3, we analyze which sensors become seeds in each corona, and we apply probabilistic tools to derive the network density which guarantees that the training process does not prematurely terminate. Finally, Section 3.3.3 presents the simulation results on the performance of the new protocol.

3.3.1 Sensor and Actor Algorithms

The proposed corona localization protocol for DC-WSANs consists of two phases: Phase I, driven by the actor and thus centralized; and Phase II in which only sensors participate.

In the first phase, the actor broadcasts for k times over successively larger radii. Specifically, as illustrated in Algorithm 1, at time slot t, with $0 \leq t \leq k-1$, the actor broadcasts beacon t using the isotropic antenna with transmitting power $w^a_{T_x} = g^{-1}(\frac{R}{k}(t+1))$, which is sufficient to reach the external radius of corona t. Once Phase I has been completed, the actor does not participate in the localization protocol any further.

When a sensor is awake in Phase I, as soon as it receives a beacon (Algorithm 2, line 7), it synchronizes itself with the actor and sets the flag *heard* (Algorithm 2, line 9). Then, it listens to the channel in order to possibly extrapolate information about its corona coordinate based on received beacons. Specifically, in Algorithm 2, a sensor that receives beacon $c = 0$ is in corona 0 (Algorithm 2, line 12). When a sensor receives for the first time (Algorithm 2, line 9) a beacon c not in the first time

Algorithm 1 Actor algorithm

```
1: procedure ACTOR-I(k)
2:     for t ← 0 to k − 1 do
3:         transmit beacon t up to corona t
4:     end for
5: end procedure
```

slot of its awake period (Algorithm 2, line 12), it realizes that it is covered by the current actor transmission up to corona c but not from the previous transmission up to corona $c-1$, since it is aware of the actor transmission pattern. Thus, the sensor learns that it belongs to corona c. Note that, whenever a sensor acquires location information from the actor, such information is correct. Therefore, sensors covered in Phase I are trained. In the rest of this paper, such trained sensors will be termed as *seeds*.

After $\frac{k}{L}$ sleep-awake periods, sensors enter Phase II. During this phase, the seeds disseminate the location information to their uncovered and awake neighbors by means of intra-cluster communication. In this way, a chain effect on the time zones starts. Seeds cover the sensors which are awake while they disseminate and which reside in a seed's transmission range. The sensors covered by the seeds, in their turn, continue the chain effect towards their awake and uncovered neighbors. This process iteratively expands until all the time zones in the corona are covered.

Observe that, if a sensor is seed in its corona, we assign to it an *index* that varies from 0 to $-(d-2)$ (Algorithm 2, line 15) and that indicates in which time slot of the awake period of the seed the actor transmitted beacon $c-1$. In other words, by means of *index*, all the seeds know the global time $c-1 = z(s) - index$ when beacon $c-1$ was broadcast. Each seed s sets its alarm clock to the local time k (Algorithm 2,

Algorithm 2 Sensor algorithm for Phase I

```
1: procedure SENSOR-I(d, L, k)
2:     covered ← false, corona ← ∞
3:     heard ← false, index ← +∞, t ← 0
4:     while wakeup ∧ t < k do
5:         for i ← 0 to d − 1 do
6:             listen to the channel
7:             if received beacon c from the actor then
8:                 t ← c
9:                 if ¬heard then
10:                    heard ← true
11:                    if ¬covered then
12:                        if c = 0 ∨ i > 0 then
13:                            corona ← c
14:                            covered ← true
15:                            index ← −(i − 1)
16:                            set alarm at time slot k and jump to Algorithm 3
17:                        end if
18:                    end if
19:                end if
20:            end if
21:            if received beacon c from a sensor then
22:                jump to line 11 of Algorithm 3
23:            end if
24:            t ← t + 1
25:        end for
26:        if ¬covered then
27:            t ← t + L − d
28:            set alarm at time slot t
29:        end if
30:    end while
31: end procedure
```

line 16), that is, at the global time $k+z(s)$ and jumps to line 1 of Algorithm 3. When a seed wakes up at time k, it enters in Phase II. Thus, seed s disseminates in Phase II at the local time $-index+1$, that is at the global time $k+z(s)-index+1 = k+c$. In this way, using $index$, all the seeds start the dissemination of the location information (Algorithm 3, line 6) to their neighbors at the same time slot.

If a sensor does not happen to become a seed in its corona, it can detect that it is in Phase II whenever it receives a beacon c not originated by the actor (recall that each beacon has a tag indicating the originator and both actor and sensors transmit on the same channel) (Algorithm 2, line 21). Thus, such a sensor jumps to Phase II at line 11 of Algorithm 3. Note that, if a sensor is not a seed, $index$ remains set to $+\infty$. Then, when a sensor becomes covered at the i-th time slot (with $0 \leq i \leq d-1$) of any awake period in Phase II (Algorithm 3, line 11), the sensor will retransmit the beacon in its neighborhood because $i > -\infty$ holds.

Finally, during both phases, a sensor not yet covered alternates between an awake period of d time slots and a sleep period of $L-d$ time slots (Algorithm 2, line 26; and Algorithm 3, line 17).

Summarizing:

Theorem 1. *Given fixed d, L, and k, the corona localization protocol requires overall $k+2L$ time slots. Each sensor stays awake for at most $2+\frac{k}{L}$ awake periods of length d, and transmit for at most d time slots.*

Our algorithm requires $O(k)$ time as the fastest algorithm presented in the literature [52]. However, the algorithm in [4] is centralized and applies to sensors that behave synchronously, while ours is semi-distributed and applies to sensors that behave asynchronously.

Algorithm 3 Sensor algorithm for Phase II

```
1: procedure SENSOR-II(d, L)
2:     t ← 0
3:     while wakeup ∧ t < 2L do
4:         for i ← 0 to d − 1 do
5:             if covered then
6:                 if i > −index then
7:                     transmit beacon corona over radius r
8:                 end if
9:             else
10:                listen to the channel
11:                if received beacon c then
12:                    corona ← c, covered ← true
13:                end if
14:            end if
15:            t ← t + 1
16:        end for
17:        if ¬covered then
18:            t ← t + L − d
19:            set alarm at time t
20:        end if
21:    end while
22: end procedure
```

3.3.2 Analysis of Localization Protocol

In this section, the proposed algorithm is mathematically analyzed from three perspectives. First, we analyze the results achieved in the deterministic Phase I, outlining which sensors become seeds in each corona, and describe the chain effect of the localization process in Phase II. Second, we apply probabilistic tools, such as results for the coupon collector's problem [57] and Chernoff bounds [57], to derive the density required to guarantee that the chain effect does not terminate before the last time zone is reached. Third, we leverage results of the previous step to compute the ratio of sensors which are covered by the positioning information originated from the seeds of Phase I. Besides the analytical evaluation of the algorithm, we make a few observations regarding the chance of mistraining and explain how some features of the algorithm limit this occurrence.

3.3.2.1 Seeds

Since each sensor in corona c has exactly one chance to get trained as a seed in time slot c, it holds:

Lemma 1. *A sensor residing in corona* 0 *becomes seed when it receives beacon* 0*, while a sensor in corona* $c \geq 1$ *becomes seed if it does not hear beacon* $c-1$ *and receives beacon* c *in the same awake period.*

Proof. By corona definition, sensors in corona c can only receive actor transmissions of radius equal to or larger than $c+1$, which transmit beacons $b \geq c$. Thus, sensors in corona $c \geq 1$ are the only ones that cannot be reached by the actor transmission of radius c, but which are reached by the transmission of radius $c+1$ with beacon c. Similarly, sensors in corona 0, are the only ones that can receive the actor transmission of radius 1 with beacon 0. □

When $c \geq 1$, since a sensor requires two time slots to be trained, we assume $d \geq 2$. Moreover:

Lemma 2. *In corona c, the seeds are the sensors that belong to time zone z such that*

$$z = \begin{cases} 0 & \text{if } c = 0 \\ [0, c-1] & \text{if } 1 \leq c \leq d-2 \\ [|c-d+1|_L, \ldots, & \\ \quad \ldots, |c-1|_L] & \text{if } d-1 \leq c \leq k-1 \end{cases} \tag{3.3}$$

Let Z_c and $|Z_c|$ denote the set of time zones whose sensors are seeds in corona c and its cardinality, respectively. Then we have $|Z_0| = 1$, $|Z_c| = c$ for $1 \leq c \leq d-2$, and $|Z_c| = d-1$ for $c \geq d-1$.

Proof. Consider a corona $c \in [d-1, k]$. By the training condition in Phase I, a sensor is trained in corona c, if it is awake while the actor transmits beacons $c-1$ and c. Since a sensor of time zone z is in its $\lfloor \frac{c-1}{L} \rfloor$-th sleep-awake period while the actor broadcasts $c-1$, such a sensor is trained if $z + \lfloor \frac{c-1}{L} \rfloor L \leq c-1 \leq z + \lfloor \frac{c-1}{L} \rfloor L + d - 2$, that is, if $|(c-1)-(d-2)|_L \leq z \leq |c-1|_L$. Moreover, in corona 0 only sensors of time zone 0 can hear the actor broadcasting corona 0, and thus become seeds, whereas in coronas $1 \leq c \leq d-2$ the seeds are sensors of time zone z such that $0 \leq z \leq c-1$. □

3.3.2.2 Chain Effect

Lemma 3. *Under suitable density property, in corona c, all sensors can be covered at the end of the interval*

$$t \in \begin{cases} [k + |c|_L, k \ + \ |c|_L + L - |Z_c| - 1] \\ \qquad \text{if } 0 \leq c \leq d-2 \text{ or } d-1 \leq |c|_L \leq L-1 \\ [k + L + |c|_L, k + L \ + \ |c|_L + L - |Z_c| - 1] \\ \qquad \text{if } c \geq L \text{ and } 0 \leq |c|_L \leq d-2 \end{cases} \tag{3.4}$$

Proof. Consider a corona c such that $d-1 \leq |c|_L \leq L-1$. Since $c \geq d-1$, by Lemma 2, time $t = k + L + |c-1|_L$ is the first time slot in which all seeds of corona c are awake. In the subsequent time slot $t+1$, while all the seeds broadcast, sensors of time zone $|t+1|_L$ wake up and listen to their neighbors in order to be trained. As discussed later, if the density is sufficiently high, each sensor of time zone $|t+1|_L$ has high probability to be in the transmission range of at least one seed; thus it will hear a corona information and will be covered. In each time slot of the interval $[k+|c|_L), \ldots, k+|c|_L)+L-|Z_c|-1]$, the chain effect continues and the time zone that wakes up is covered. Since $|Z_c|$ sensors were seeds in corona c, and $L-|Z_c|$ are trained in Phase II, all the different time zones are trained by the end of the claimed interval. A similar reasoning can be repeated for coronas $c \geq L$ such that $0 \leq |c|_L \leq d-2$, observing that one has to wait the second awake period of the last seed $|c-1|_L$ to have all the seeds awake simultaneously. □

It is easy to see from Eq. (3.4), that Phase II terminates within time slot $k+2L$.

3.3.2.3 Density

To ensure that the chain effect is not terminated prematurely, the localization protocol relies on the presence in the area covered by sensor s_i of at least one sensor s_j of time zone $z(s_i) < z(s_j) \leq z(s_i) + (d-1)$. Thus, it is important to identify the density Λ of the Poisson point process behind the sensor network for which this condition holds. We first derive the result for $d = 2$, for which the constraint simplifies to $z(s_j) = z(s_i) + 1$, and then we extend the result to the case $d > 2$.

In our model, in the transmission area πr^2 of each sensor, there are m neighbors which choose their time zone uniformly and independently at random in the interval $[0, L-1]$. Since sensors are distributed in the area uniformly at random, the actual number of neighbors in such area is not deterministically set but given by a random variable Y that follows a Poisson probability distribution of mean $\mu_Y = E[Y] = \Lambda\pi r^2$ as stated in Eq. (3.2). Moreover, the neighbors of time zone z in the transmission area have a Poisson distribution of mean $\mu_{Y_z} = E[Y_z] = \frac{\Lambda\pi r^2}{L}$. Clearly, the chain effect does not terminate prematurely if all the time zones are present in each transmission area. The trivial bound $m = L$ for the neighbors in a sensor transmission area does not work because it only guarantees an average of one (and not at least one) sensor for each time zone.

The problem of finding the required number m of neighbors so that there is, with high probability, at least one sensor of each time zone is equivalent to the *coupon collector's problem* (CCP) in which there are L different kinds of coupons and the objective is to identify the number of trials m that are to be performed in order to collect a sample of each kind with high probability [57]. The solution to the

CCP states that the following number of trials is necessary to achieve the given objective [57]:

$$m = L \ln L + O(L) \tag{3.5}$$

Therefore:

Theorem 2. *Fixed $p \to 1$, the number m_{CCP} of neighbors in a sensor transmission area of radius r which guarantees that with high probability p there is at least one sensor of each time zone is:*

$$m_{CCP} = L \ln L + c_p L \tag{3.6}$$

where

$$c_p = -\ln(-\ln p) \tag{3.7}$$

Proof. Eq. (3.5) provides the solution to the number m of neighboring sensors that is required in order to guarantee that the chain effect does not terminate prematurely in the domain of the DC-WSAN localization protocol. Since an actual number m is needed to apply the proposed algorithm, a constant $c_p \geq 0$ is to be derived such that each sensor has got all time zones among its neighbors with a required high probability $p \to 1$. To this extent, X is defined as the random variable representing the number of trials required to collect all coupons. Then, for any constant $c_p \in \mathbb{R}$ the following result holds [57]:

$$Pr[X > m = L(\ln L + c_p)] \approx 1 - e^{-e^{-c_p}}$$

Thus, a specific value of constant c_p can be computed from the requested probability $p = 1 - Pr[X > m = L(\ln L + c_p)]$ of having all time zones among the neighbors of sensor s by the following formula:

$$c_p = -\ln(-\ln p)$$

By substituting c_p into Eq. (3.5), the required number of neighbors $m_{CCP} = L \ln L + c_p L$ is derived. □

From Eq. (3.6), a density $\Lambda_{CCP} = \frac{L \ln L + c_p L}{\pi r^2}$ can be computed for the Poisson point process modeling the sensor network. However, density Λ_{CCP} does not guarantee that there are exactly, or at least, m_{CCP} sensors in a sensor's neighborhood as this is only the expected value $E[Y]$ of the random variable Y. Thus, further analysis is necessary to provide additional constraints to guarantee that there is indeed a sufficient number of neighbors, and Chernoff bounds can provide a solution to this problem [57]. Applying Chernoff bounds to the distribution of the random variable Y, we obtain:

Theorem 3. *Given fixed $p, q \to 1$, a Poisson point process of density*

$$\Lambda_{CB,CCP} = \frac{c_q(L \ln L + c_p L)}{\pi r^2} \tag{3.8}$$

where

$$c_q = \left(1 - \frac{\ln(1-q)}{m_{CCP}}\right) \pm \sqrt{\left(\frac{\ln(1-q)}{m_{CCP}} - 1\right)^2 - 1}, \quad c_q > 1 \tag{3.9}$$

and

$$c_p = -\ln(-\ln p)$$

provides that, with high probability q, there are at least $m_{CCP} = L \ln L + c_p L$ sensors in each sensor's neighborhood and that each neighborhood has, with probability p, at least one sensor per time zone.

Proof. As far as the left tail of the random variable Y is concerned, the following bound applies [57]:

$$Pr[Y \leq \mu_Y - \delta\mu_Y] \leq e^{-\frac{\delta^2 \mu_Y}{2}} \qquad \textit{with } 0 < \delta \leq 1.$$

Since our objective is to find $\mu_Y = c_q m_{CCP}$ such that there is a given high probability $q \to 1$ that there are at least m_{CCP} sensors in a sensor's neighborhood, we can retrieve the constant factor $c_q > 1$ by solving the following system of equations:

$$\begin{cases} 1 - q = e^{-\frac{\delta^2 \mu_Y}{2}} \\ m_{CPP} = (1 - \delta)\mu_Y \end{cases} \tag{3.10}$$

From simple algebraic manipulation of Eq. (3.10), one can derive the claimed value of c_q and thus the desired density $\Lambda_{CB,CCP}$. □

The result in Eq. (3.5) and Eq. (3.8) can be extended to the case $d > 2$. In this situation the chain effect propagates as long as there is among the neighbors of sensor s_i at least one sensor s_j of time zone $z(s_i) < z(s_j) \leq z(s_i) + (d-1)$. Since there is not only one time zone that fulfills this requirement, the coupon collector's problem can be restated to gather at least one sample of $\frac{L}{d-1}$ different kinds of coupons, thus lowering the network density requirement. However, this does not guarantee that the chain effect propagates as the following example demonstrates.

Let Z_i^d and Z_{i+1}^d be respectively the set of time zones in the i^{th} and $(i+1)^{th}$ groups of size $d-1$. The CCP solution with $\frac{L}{d-1}$ states that there is one sensor $s_i \in Z_i^d$ of time zone $z(s_i) \in [j-d, j-1]$ and one sensor $s_{i+1} \in Z_{i+1}^d$ of time zone $z(s_{i+1}) \in [j, j+(d-1)]$. Nonetheless, sensors s_i and s_{i+1} can be of time zones $z(s_i) = j - d$, $z(s_{i+1}) = j + (d-1)$, and thus be more than $d-1$ time zones apart (*e.g.*, $z(s_{i+1}) - z(s_i) = j + (d-1) - (j-d) = 2d - 1 > d - 1$) so that the chain effect terminates. Therefore, the group size has to be set so that the smallest time zone in set Z_i^d can cover all time zones in set Z_{i+1}^d, which yields a set size $G = |Z_i^d| = \lfloor \frac{d}{2} \rfloor$. Finally, when $d > 2$, the general formula for the number of neighbors m can be rewritten as

$$m_{CPP} = \frac{L}{G} \ln\left(\frac{L}{G}\right) + O\left(\frac{L}{G}\right) \tag{3.11}$$

		$q = 0.99$		$q = 0.95$	
		$d = 2$	$d = 4$	$d = 2$	$d = 4$
	c_p	c_q		c_q	
$p = 0.99$	4.600149	1.510192	1.841638	1.5103	1.6407
$p = 0.90$	2.2503673	1.665466	2.172956	1.5112811	1.8799505
$p = 0.85$	1.8169607	1.711033	2.280284	1.5449350	1.9564874

Table 3.1. Values of c_p and c_q when $L = 8$.

and the density as

$$\Lambda_{CB,CCP} = \frac{c_q \left(\frac{L}{G} \ln\left(\frac{L}{G}\right) + c_p \frac{L}{G}\right)}{\pi r^2} \tag{3.12}$$

In Table 3.3.2.3, the values of c_p and c_p are computed when $L = 8$, $d = 2$ or 4, $p = 0.99$, 0.9 or 0.85, and $q = 0.99$ or 0.95.

3.3.2.4 Coverage

We now turn to the analysis of the coverage area, that is the ratio of sensor locations that are covered by the algorithm. Recall that the set of covered sensors includes both the ones that are successfully trained and the ones that are mistrained.

In order to perform such analysis, let us call sensor s in corona $c(s)$ a *reference seed* if s becomes seed in the second time slot of its awake period. Moreover, the time zone $z(s)$ of s is referred to as the *reference time zone*. We now index the time zones with respect to the reference time zone. Precisely, the reference time zone gets index $j = 0$, and time zone z gets index $z - z(s)$. Thus, negative indices are attributed to the $d - 2$ time zones corresponding to seeds in corona $c(s)$ preceding the reference seed, whereas time zones following the reference seed feature positive indexes. Finally, we define the *chain effect coverage function* $f(j)$ as the coverage area of time zone $|z(s) + j|_L$ of index j with $f : [-(d-2), L-(d-2)) \to [0, 1]$. This definition of the chain effect coverage function makes the following analysis independent of the actual

corona, as far as it has $d-1$ seeds, since the indexes are accordingly shifted in each corona. As it will be apparent later, the proposed analysis is slightly more optimistic for coronas $0 \leq c \leq d-1$ since there is a smaller number of seeds in such coronas as stated by Lemma 2. Thus, a comprehensive analysis of both the first and the second phase can be performed by means of the defined chain effect coverage function.

For simplicity, the chain effect coverage function is first derived for the case $d = 2$, where $f : [0, L-1] \rightarrow [0, 1]$ because in each corona c (except corona 0), only the time zone $|c-1|_L$ becomes seed and gets index 0. Given the deterministic nature of the first phase of the algorithm, we can state that the whole area is covered as far as seeds are concerned, from which we obtain

$$f(0) = 1 \tag{3.13}$$

According to previous research work on Poisson point processes [56], the area being covered by seeds is

$$f(1) = 1 - e^{-\lambda \pi r^2 f(0)} \tag{3.14}$$

where $\lambda = \frac{\Lambda}{L}$ is the density of seeds within a time zone. By iterating over the time zones, the area which is covered can be computed for each index j as follows:

$$f(j) = \begin{cases} 1 & if\ j = 0, \\ 1 - e^{-\lambda \pi r^2 f(j-1)} & if\ 1 \leq j < L. \end{cases} \tag{3.15}$$

Note that the chain effect coverage function $f(j)$ can be intended both as the ratio between the covered area and the total area, and the ratio between the number of covered sensors in time zone of index j and the total number of sensors in time zone of index j, since sensors are deployed uniformly at random. Thus, we generically refer to the covered ratio in the mathematical analysis and the discussion of simulation results.

When $d > 2$, several time zones of sensors are awake and ready to eventually cover the set of awakening ones. Thus, the result in Eq. (3.15) can be generalized to any value of d by substituting the single term $f(j-1)$ with a sum $\sum_{i=j-(d-1)}^{j-1} f(i)$ of $d-1$ terms as follows:

$$f(j) = \begin{cases} 1 & if\ -(d-2) \leq j \leq 0, \\ 1 - e^{-\lambda \pi r^2 \sum_{i=j-(d-1)}^{j-1} f(i)} & if\ 1 \leq j < L-(d-2). \end{cases} \tag{3.16}$$

The chain effect coverage function $f(j)$ as defined in Eq. (3.16) applies when there is, with high probability, a sensor of each time zone in the neighborhood as guaranteed by Eq. (3.6). However, if the number of neighbors is reduced by considering groups of size G as in Eq. (3.11), the sum of $d-1$ terms in Eq. (3.16) collapses into one single term as the density in Eq. (3.12) guarantees the presence of at least one time zone among the $d-1$ preceding time zones $[j-(d-1), j-1]$. Thus, in case of reduced density as in Eq. (3.12), the chain effect coverage function defined in Eq. (3.15) should be used to model the performance of the algorithm.

Although a closed-form equation for the chain effect coverage function $f(j)$ would be useful, we could not find it so far. Thus, a numerical evaluation of $f(j)$ has been carried out along with that of the *cumulative coverage function* $F(j)$ defined as:

$$F(j) = \frac{\sum_{i=-(d-2)}^{j} f(i)}{L} \qquad for\ -(d-2) \leq j \leq L-(d-1) \tag{3.17}$$

which evaluates the comprehensive coverage of the algorithm. In the following, we denote with $F^* = F(L-(d-1))$ the final coverage of the algorithm.

In Figure 3.5, the chain effect coverage function f and the cumulative coverage function F are plotted for $d = 2$, for different values of L and for their corresponding density $\Lambda_{CB,CCP}$ computed from Eq. (3.8) with $c_p = 0$ and $c_q = 1$. It is worthwhile

observing that for all values of L the chain effect coverage function approaches a horizontal asymptote within few time zones from the seed. This represents an important result as we can claim that the coverage does not degrade as sensors belonging to time zones further away from the seed awake.

The plots also show a coherent behavior between density and coverage. A smaller number of time zones L, and thus lower density, implies a smaller number of sensors and also a lower area coverage. However, one has to bear in mind that Eq. (3.17) obtains an upper bound for the ratio of sensors to which data are disseminated. This is due to the fact that the method behind Eq. (3.16) assumes that there is exactly the expected number of sensors within a sensor's communication range. Clearly, this is not the case due to the random Poisson point process, and thus a sensor might be within two seeds' communication range, and another have none.

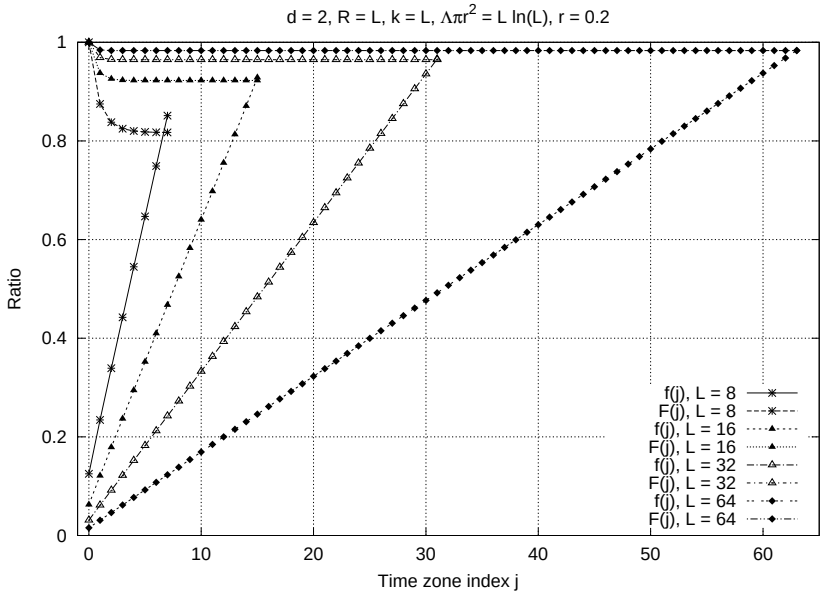


Figure 3.5. Chain effect coverage function $f(j)$ and cumulative coverage function $F(j)$ for constant d and different values of L.

If a better performance in terms of coverage is requested, while keeping the same density, the value of d can be tuned accordingly. Figure 3.6 portrays results for constant $L = 8$ and increasing d demonstrating how an increase from $d = 2$ to $d = 3$ can bring about a significant improvement on the area coverage from 0.84 to slightly more than 0.98. This shows a trade off between area coverage and sensor awake time.

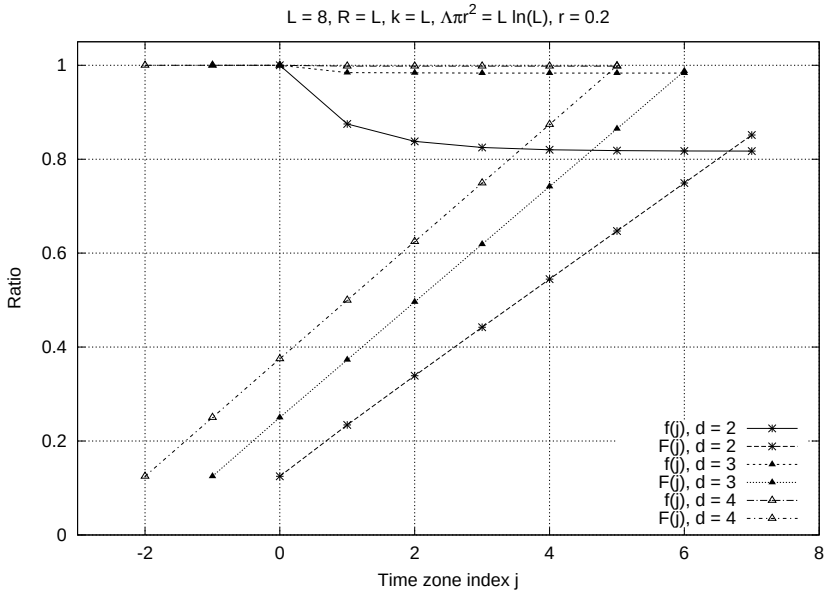


Figure 3.6. Chain effect coverage function $f(j)$ and cumulative coverage function $F(j)$ for constant L and different values of d.

3.3.2.5 Mistraining

While in Section 3.3.2.4 general analytical results were presented for the area coverage provided by the algorithm, similar results for the mistrained sensors are more difficult to derive due to the random deployment of sensors and the complexity of the communication patterns among them. However, few observation can be made about this issue. First of all, seeds (*i.e.*, sensors covered in the first phase) are all correctly trained since we make the assumption that the actor reaches perfectly shaped circles

while it broadcasts its beacons. In the second phase, the worst case scenario arises when sensors in corona c are awake and listening and only sensors in corona $c-1$ are broadcasting. The localization algorithm addresses this problem by stating that seed s_i of time zone $z(s_i) \in [|c(s_i) - (d-1)|_L, |c(s_i) - 1|_L]$ in corona $c(s_i)$ starts broadcasting in the second phase at time slot $t = |c(s_i)|_L$ as enforced by conditional statement on line 6 of Algorithm 3.

Thus, a sensor in corona c can become mistrained only if it is awake at time $t = L + c - 1$ and close enough to the border with corona $c-1$ so that it can receive messages from the seeds in corona c. Indeed, from the next time slot $t = L + c$, the seeds in both corona $c-1$ and corona c broadcast the corona they have learnt and since such coronas are different, the uncovered sensors cannot learn. In conclusion, the use of the variable *index* to synchronize the seed transmissions in each corona during Phase II and the fact that each neighborhood has at least one seed (see Eq. (3.10)) contribute to keeping the probability that a sensor becomes mistrained low. However, to further control the impact of the mistrained sensors on the network behavior, one can introduce a third phase that lasts L time slots. During such a phase, while awake, the trained sensors repeatedly devote one time slot to broadcast and one time slot to listen. Specifically, trained sensors with even (resp., odd) time zones broadcast their learnt corona in the even (resp., odd) time slots and listen to their neighbors in the odd (resp., even) time slots. If during all the time slots of the third phase a sensor receives concordant messages, coinciding with its own corona, it apprehends to be trained correctly and can eventually participate to the time zone leader election[3]. Instead, if during at least one time slot of the third phase a trained sensor receives

[3]Note that a sensor cannot discover to be mistrained if all the sensors in a neighborhood have learnt the same faulty corona. Nonetheless, this event happens quite rarely if the network density satisfies Eq. (3.10).

discordant messages, it recognizes either to be mistrained or to be very close to the corona border. In both cases, it does not participate to the leader election protocol and eventually performs only local computations.

3.3.3 Simulation Results

A simulator was developed to study the performance of the proposed algorithm and validate the mathematical analysis presented in Section 4.3. The simulator employed to perform the following experiments was implemented in C++. It relies on the Boost C++ Library (http://www.boost.org) and the GNU Scientific Library (http://www.gnu.org/software/gsl) to manage the network graph and generate samples from random distributions, respectively. Although a large set of configurations were simulated, including experiments demonstrating how densities derived from $p, q \to 1$ yield a coverage of up to 0.95 for $d = 2$ and $L = 8$, due to lack of space the discussion in this section is limited to the experiments with $c_p = 0$ and $c_q = 1$ to demonstrate that the chain effect is actually ignited and kept alive already when the minimal density requirements derived in Section 3.3.2.3 are fulfilled. Second, results are presented that verify the algorithm coverage capability. Finally, the issue of mistraining is addressed by presenting experimental results that show how a very small ratio of sensors is actually mistrained.

Experimental results for chain effect coverage and cumulative coverage functions are presented for $L = 8,\ 16$ in Figure 3.7. Such results follow the same pattern of the analytical function derived in Sections 3.3.2.3 and 3.3.2.4, though approximating slightly lower asymptotes. This difference between simulation results and analytical functions is due to the asymmetric random nature of the sensors and network model. In fact, although the random variable Y representing the number of neighbors has expected value $E[Y] = L\ln(L)$ in the optimal case, the actual values vary around

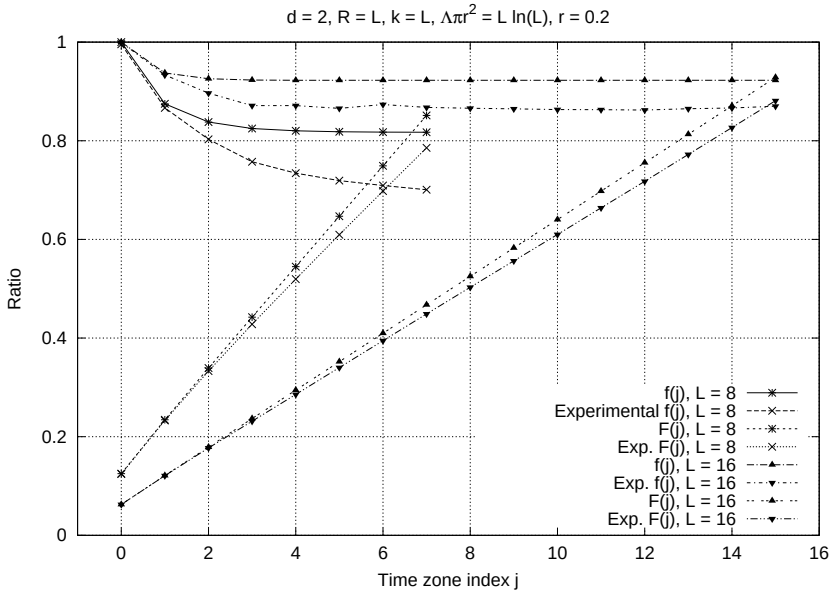


Figure 3.7. Experimental results, analytical chain effect coverage function and cumulative coverage function for constant d and different values of L.

the mean, and sensors with more neighbors than the required number to be covered do not compensate for sensors with less neighbors (*i.e.*, a sensor with twice as many neighbors is not covered twice, thus not making up for a sensor with no neighbors which is not covered). Therefore, the analytical result is always better than the simulation except when all sensors are covered by the algorithm.

Finally, as discussed in Section 3.3.2.4, the algorithm performance can be improved by increasing the awake time d. Experimental results for $L = 8$ are depicted along with the cumulative coverage function in Figures 3.8 and 3.9. It is worthwhile observing that the ratio of mistrained sensors over the total number of sensors (which is represented by the difference between experimental $F(L-(d-1))$ and the ratio of sensors trained in Phases I and II) is constant across different values of d. Furthermore, we were also able to verify that all such instances of mistrained sensors feature an absolute error of 1, that is, sensors in corona c are mistrained to belong to corona

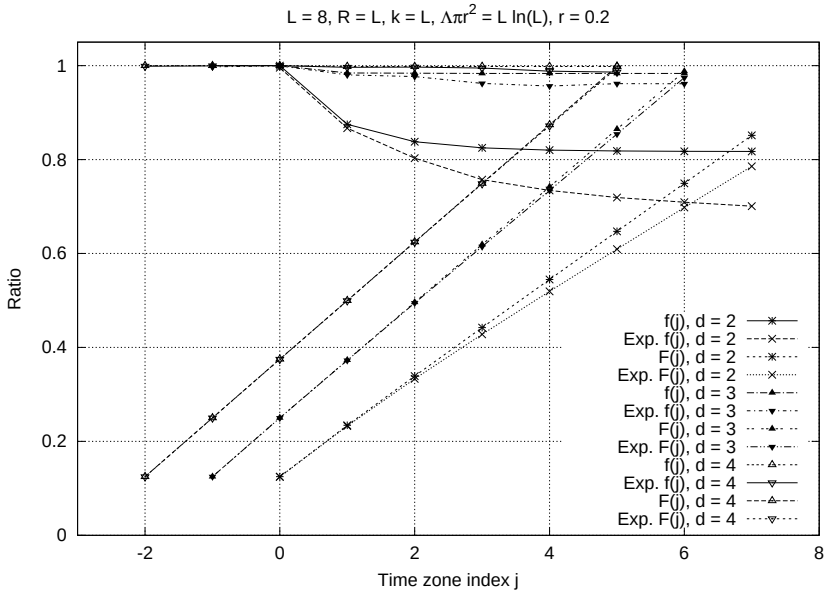


Figure 3.8. Experimental results and analytical chain effect coverage function and cumulative coverage function for constant L and different values of d.

$c-1$ or $c+1$ but not beyond, and their occurrence is limited to the area within a distance of 0.4 from the border, when $r = 0.2$ and $\frac{R}{k} = 1$.

Once the performance of the algorithm featuring an optimal density is analyzed, it is worthwhile studying what is the effect of a decay of the network density on the chain effect coverage function, assuming that the chain effect is not prematurely terminated due to the lower density itself. An error term can be added to the definition of Λ as follows:

$$\Lambda = \frac{L(\ln L)^{1-\epsilon}}{\pi r^2} \qquad 0 \leq \epsilon \leq 1 \tag{3.18}$$

Figure 3.10 shows the analytical and experimental chain effect coverage function when $\Lambda = L$ (or, $\epsilon = 1$). The asymptotes in Figure 3.10 are radically different from those observed in Figure 3.7.

Finally, Figure 3.11 portrays coverage as a function of density decay ϵ. While

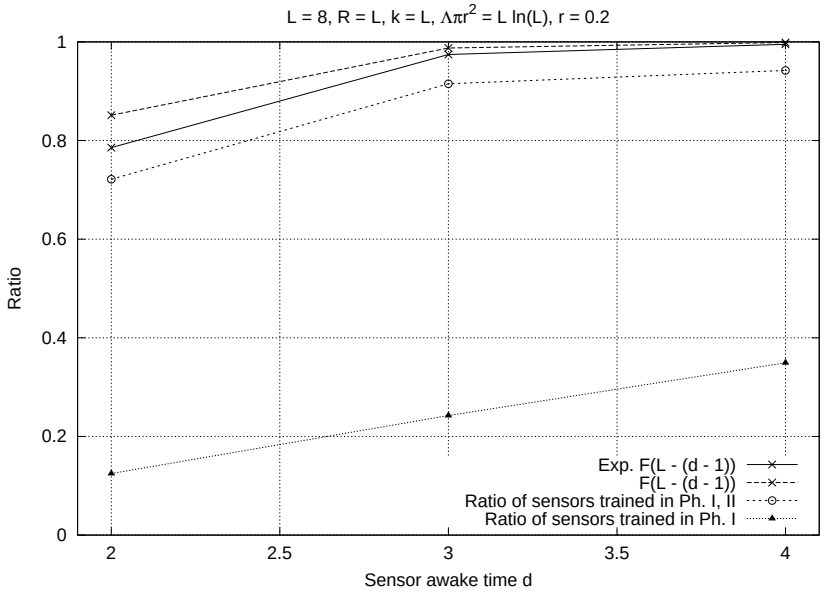


Figure 3.9. Experimental results, including ratio of seeds, trained, and mistrained sensors, and analytical cumulative coverage for constant L and different values of d.

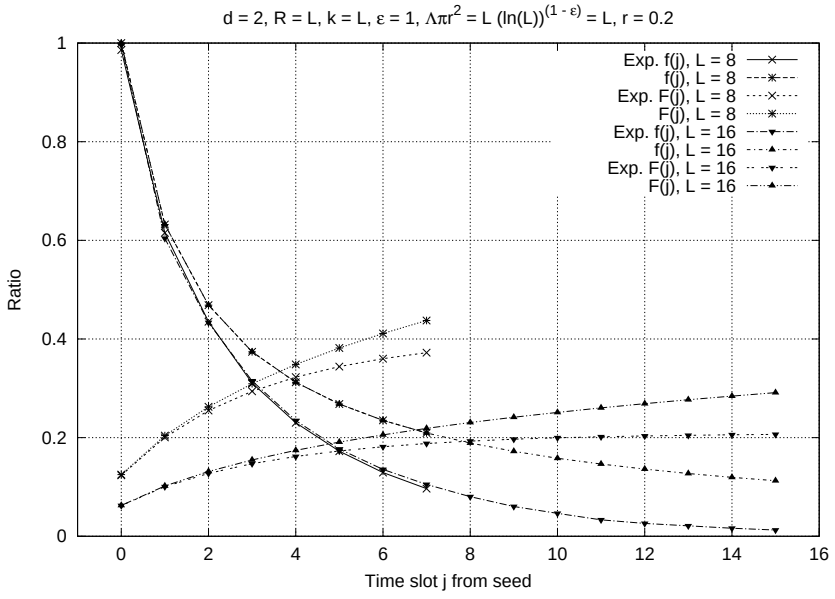


Figure 3.10. Experimental results and analytical chain effect coverage function and cumulative coverage function for suboptimal density for constant d and different values of L.

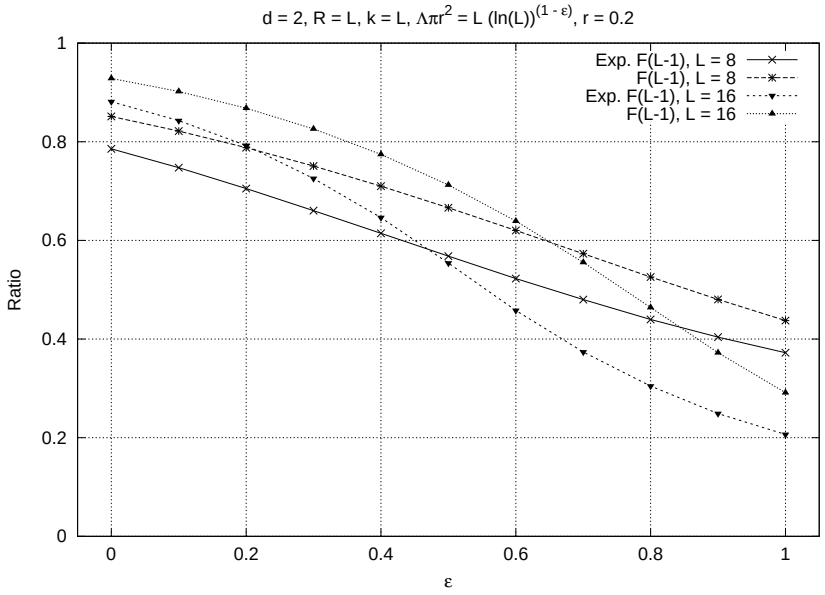


Figure 3.11. Experimental results and cumulative coverage $F(L-1)$ as a function of density decay ϵ.

instances with a larger number of time zones L perform better when the density constraint is upheld, they feature a steeper degradation compared to solutions with a smaller number of time zones L until they eventually perform worse, as indicated by the pattern of intercepts in the right-hand side of the plots.

3.4 Summary

In this chapter, we addressed one of the fundamental problems arising in the context of DC-WSAN when mobile actors set up actor-centric networks: localization. We investigated a virtual organization of an actor-centric subnetwork, consisting of a single actor and of many sensors that follow periodic sleep-awake schedules. We imposed a generalized polar coordinate system whose goals are *i)* to provide a coarse-grained location that partitions the network in clusters; and *ii)* to easily support a geographic routing algorithm as well as a leader election algorithm.

To impose the coordinate system, we propose a localization algorithm that consists of a centralized phase and a distributed phase. At the end of the algorithm execution, sensors have learned the polar coordinate of the region they belong to. Our new contribution to localization is the analysis of the sensor coverage with respect to the sensor density. We applied well-known stochastic analysis tools such as coupon collector's problem and Chernoff bounds to derive bounds on the density required to achieve coverage of all sensors with high probability. The analytical bounds were verified with experimental results from a software simulator that we developed. The simulation results demonstrate that very good results (in terms of final coverage ratio and chain effect without interruptions) can be achieved already when $c_p = 0$ and $c_q = 1$.

CHAPTER 4

ENERGY-EFFICIENT MARKOV CHAIN-BASED DUTY CYCLING SCHEMES

Since WSNs consisting of several hundreds or thousands of wireless nodes are deployed in almost every domain of our life to solve many different problems, the design of sustainable and green solutions for wireless nodes will have a large impact in terms of resource consumption. Energy consumption is an overarching problem for any operation in a WSN, because wireless sensor nodes usually have a limited energy budget. This is due to the deployment environment (for example, sensor nodes embedded in a wall or air-dropped on a difficult terrain), or high density deployment such that replacing batteries on hundreds or thousands of devices is infeasible, no matter how accessible they are. Even if the sensor nodes harvest energy from the environment, such as using tiny solar panels, the power supply is limited and sensor nodes cannot operate at maximum speed at all times [58].

In order to increase the network life-time, wireless sensor nodes switch between active and dormant states. The ratio of time during which a sensor is awake is called *duty cycle*. Transitions between the active and dormant states cost time and energy, and thus it is important to minimize the frequency of switching. In most duty cycling schemes, sensor nodes do not decide on their next state step by step, but rather compute the working schedules ahead of time during an initialization phase. Sensor nodes can exchange information with neighboring nodes in order to generate working schedules that improve network connectivity and latency. Latency can also be reduced by having shorter time slots. To reduce the overhead due to coordination, partially randomized duty cycling schemes have been proposed [59].

In a randomized duty cycling scheme, each sensor node randomly generates a working schedule that yields the required duty cycle. The random variables that define the state at each time slot are usually considered as independent and identically distributed (*i.i.d.*). In partially randomized schemes, the random working schedules can be exchanged among neighboring nodes, so that the latency can be reduced. For instance, in the data dissemination protocol proposed in [59], wireless sensor nodes can use information about their neighbors' working schedules to anticipate message forwarding. Since wireless nodes can turn into active state beyond their working schedules, the *aggregate duty cycle* (θ), *i.e.,* the actual posterior duty cycle, is usually greater than the *working schedule duty cycle* (μ), *i.e.,* the planned prior duty cycle. The difference between the working schedule and aggregate duty cycles depends on the number of neighbors and their schedules.

The *time slot length* (T) has an impact on the *time and energy efficiency* (η) of the duty cycling scheme. The shorter the time slot, the more significant is the overhead in terms of time and energy due to operations such as opening or closing radio connections, when a sensor node transitions from and into deep sleep mode (or dormant state). The time slot length also affects the *connection delay* (δ) (or latency) after which two neighboring nodes are simultaneously active, as well as the *connection duration* (ω). The longer the time slot, the higher the connection delay and connection duration. This is because these metrics are linear functions of the time slot length, for a given working schedule.

The goals of lower connection delay and higher efficiency clearly pull the knob controlling the time slot length in opposite directions. The former requires shorter time slots, while the latter requires longer ones. Nonetheless, there exists a solution that can improve the performance along either dimension without affecting the other.

In this chapter, we propose a randomized duty cycling scheme that reduces the connection delay and maintains a comparable level of time and energy efficiency with a constant aggregate duty cycle, and thus constant energy consumption[1]. Alternatively, time and energy efficiency can be improved, yet keeping an almost constant connection delay. Our solution is based on the combination of shorter time slots (to achieve lower delay) and a Markov chain (to achieve higher efficiency). Shorter time slots yield schedules that pack more time slots in the same period of time, so that neighboring nodes are more likely to be simultaneously active earlier on. A Markov chain generates a schedule whose active time slots are more likely to be consecutive, thus yielding longer active time instances, which in turn correspond to higher time efficiency. Our novel Markov chain-based solution expands the design space of randomized duty cycling schemes by adding a third dimension, namely memory, represented by a *memory coefficient* (γ), to the two other dimensions, namely time slot length and working schedule duty cycle. We derived mathematical expressions for the aggregate duty cycle, connection delay, connection duration, and time efficiency in terms of the working schedule duty cycle, time slot length, and memory.

Experiments on Sun SPOTs [62] validate our analytical model and show that the expected connection delay can be reduced by at least 31.54%, while keeping a time efficiency of 0.9563, as opposed to a 2.48% reduction in a scheme using *i.i.d.* random variables. In turn, this corresponds to a saving between 312-536 mJ per minute per wireless node. The improvement in terms of efficiency is higher when considering more realistic (hence time and energy consuming) set up and tear down operations, such as MEMS sensor warm up and neighbor discovery.

[1]The research work presented in this chapter has been published in the Proc. of the 31st IEEE International Conference on Distributed Computing Systems (ICDCS 2011) [60], and the ACM Journal on Emerging Technologies in Computing (JETC) [61].

The rest of the chapter is organized as follows. Section 4.1 describes the concepts behind our randomized duty cycling scheme and makes the case for a Markov chain-based solution. The randomized duty cycling scheme is presented in Section 4.2 with its mathematical analysis in Section 4.3, while experimental results are discussed in Section 4.4. We review the related work on duty cycling schemes for WSNs in Section 4.5 and draw our conclusions in Section 4.6.

4.1 Motivation and Preliminary Experiments

Energy consumption is one of the most important aspects to achieve sustainable and green computing systems. In WSNs, energy consumption minimization is also required because of the limited energy budget of battery-powered wireless nodes. The *aggregate duty cycle* is the ratio of time spent in normal mode, and thus is a metric directly related to the energy consumption of a wireless sensor network.

An important metric of any duty cycling scheme is the *connection delay*. While in the dormant state, the connection between a sensor node and its neighbor is temporarily interrupted until both sensors switch into active state, thus introducing a delay in the communication. As such, the vast majority of research on duty cycling schemes aims at improving this metric (also referred to as *sleep latency*).

The *connection duration* is the time during which a pair of neighboring sensors are active simultaneously and uninterruptedly. As such, this metric is directly related to the connection delay, and thus often ignored. The *time* (*energy*) *efficiency* is defined as the ratio of the time (energy) dedicated to the wireless sensor network application over the total amount of time (energy) spent in normal operation mode. Given that a sensor node incurs into a time (energy) overhead when switching from and into deep sleep mode (*e.g.*, opening or closing radio connections, warming up sensors), the time (energy) efficiency is less than 1.

We argue that the time and energy efficiency of a duty cycling scheme are also important metrics. Interestingly, they are the subject of much attention in deep sleep mode operation for larger systems such as laptops and mobile phones [63]. However, to the best of our knowledge, they have not received much attention at all in the context of duty cycling schemes for WSNs. In this section, we present preliminary results from real experiments on the connection delay and time efficiency that motivated us to develop duty cycling schemes that bring the time efficiency metric to the foreground.

4.1.1 Randomized Scheme Model

For our research on the performance of randomized schemes, we refer to the model depicted in Fig. 4.1. In this model, each wireless node *(i)* generates its own random working schedule, *(ii)* synchronizes its clock with neighboring wireless nodes (*e.g.,* using the Flooding Time Synchronization Protocol in [64]), and then *(iii)* operates switching between dormant and active states according to its own working schedule. When the wireless node has gone through all the time slots in the working schedule, it restarts the process generating a new working schedule, or stops depending on the application.

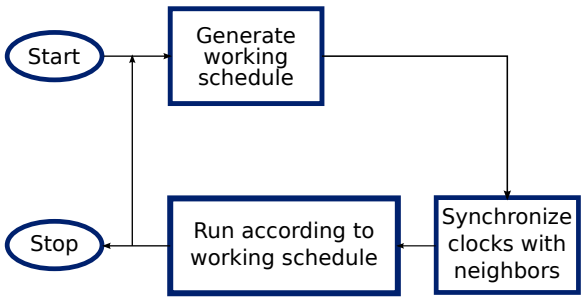


Figure 4.1. Block diagram for randomized scheme operation.

4.1.2 Aggregate Duty Cycle

The working schedule duty cycle μ might not represent the actual duty cycle, because the wireless node might switch into active state at additional times depending on the randomized scheme. For instance, this is the case for the partially randomized scheme presented in Section 5.1, where a wireless node can switch to active state in order to anticipate message forwarding to a neighbor. Therefore, we provide this definition for the aggregate duty cycle[2].

Definition 3 (Aggregate duty cycle)**.** *Given a duty cycling scheme, the aggregate duty cycle θ is the ratio between the actual time spent in the active state and the total time.*

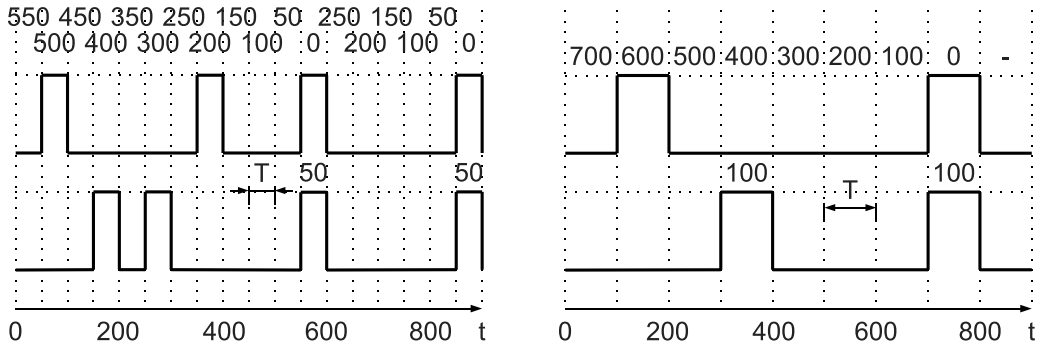
Clearly, for a randomized scheme as the one depicted in Fig. 4.1 the aggregate duty cycle θ is equal to the working schedule duty cycle μ, as confirmed by the results of our experiments on Sun SPOT sensors [62] plotted in Fig. 4.3(a). In Section 5.2.2, we show that this is not the case for partially randomized schemes where wireless nodes can switch into active state beyond their working schedules.

4.1.3 Connection Delay

Figures 4.2(a) and 4.2(b) depict working schedules of two nodes for different time slot lengths. Although both systems yield equivalent duty cycles of $\frac{4}{20} = 0.20 = \frac{2}{10}$, the nodes in Fig. 4.2(a) are likely to be simultaneously active earlier than the nodes in Fig. 4.2(b). Based on the above intuition, we define the connection delay as follows.

[2]In the literature this is simply referred to as the duty cycle. We add the adjective "aggregate" to distinguish it from the working schedule duty cycle μ.

Definition 4 (Connection delay)**.** *Given a pair of sensors, the connection delay δ is the time interval between the current time slot and the first time slot at which both sensors are active.*

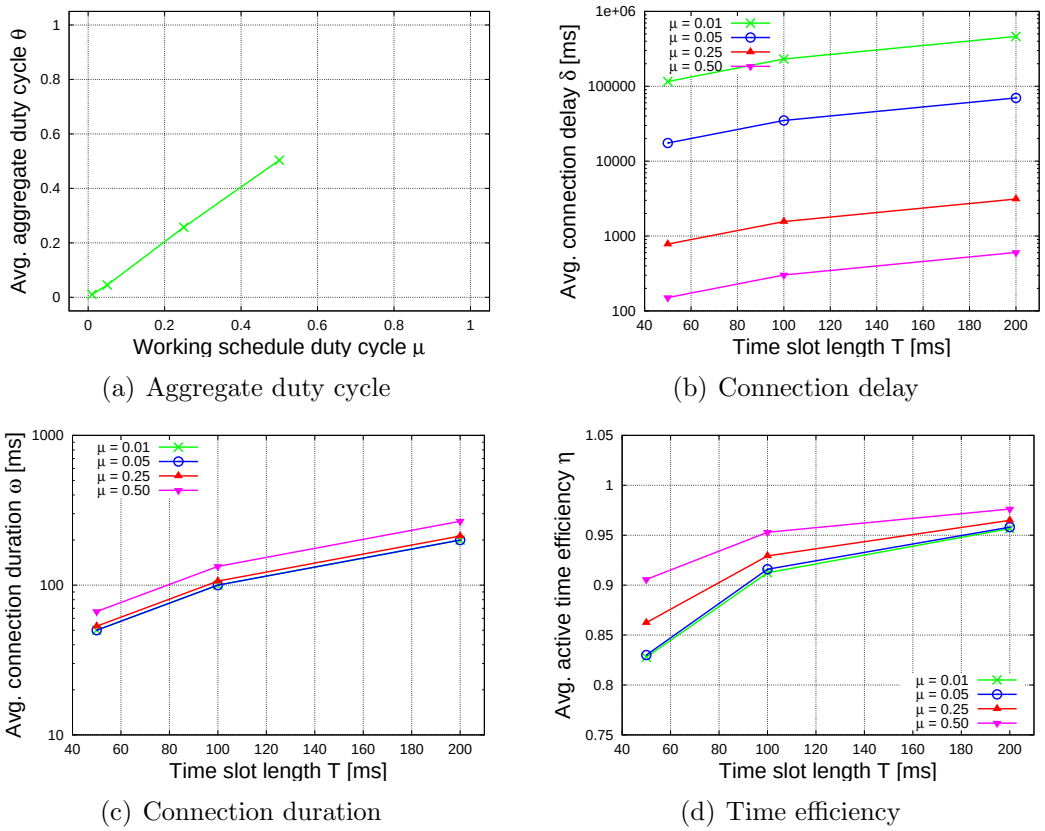


(a) Short time slot

(b) Long time slot

Figure 4.2. Sample random working schedules for different time slot lengths ((a) $T = 50$ and (b) $T = 100$ ms), but equal duty cycle $\mu = 0.20$, and thus equal energy consumption. Connection delays at each time slot are depicted on the top, while connection durations are between the schedules. Shorter time slots yield better connection delay ($\delta = \frac{550+500+\cdots+50+0+250+\cdots+50+0}{18} = 225$ ms when $T = 50$ ms, as opposed to $\delta = \frac{700+600+\cdots+100+0}{8} = 350$ ms when $T = 100$ ms). However, longer time slots yield better connection duration ($\omega = 100$ ms when $T = 100$ ms, as opposed to $\omega = 50$ ms when $T = 50$ ms).

We performed experiments on Sun SPOT sensors to measure the connection delay for different time slot lengths. The results plotted in Fig. 4.3(b) validate our intuition that shorter time slots yield smaller average connection delay. With respect to Fig. 4.3(b), for a duty cycle $\mu = 0.05$, the connection delay goes from $\delta = 69,896$ ms when $T = 200$ ms, to $\delta = 17,474$ ms when $T = 50$ ms. Thus, it is important that the time slot length be minimal, in order to increase the likelihood of an early connection between neighboring sensor nodes. Furthermore, for a given time slot length, the expected connection delay is smaller for sensors with higher duty cycles.



(a) Aggregate duty cycle

(b) Connection delay

(c) Connection duration

(d) Time efficiency

Figure 4.3. (a) Aggregate duty cycle vs. working schedule duty cycle; and (b) connection delay, (c) connection duration, and (d) time efficiency vs. time slot length for different working schedule duty cycles in a randomized scheme, when the working schedule is generated with independently and identically distributed (*i.i.d.*) random variables .

This simple experiment demonstrates that shorter time slots provide a better performance in terms of connection delay of the wireless sensor network. However, there exists a lower threshold for the time slot length in order for the sensor node to operate. This threshold depends on the communication (*e.g.,* MAC layer), sensing (*e.g.,* MEMS and analog/digital conversion), and processing features of the sensor node.

4.1.4 Connection Duration

In a WSN, the time interval during which a pair of neighboring nodes are simultaneously and uninterruptedly active is another important metric. In fact, the impact of headers in communication protocols can be reduced by transmitting longer messages, thus improving the overall performance of the protocol. We provide this definition for connection duration.

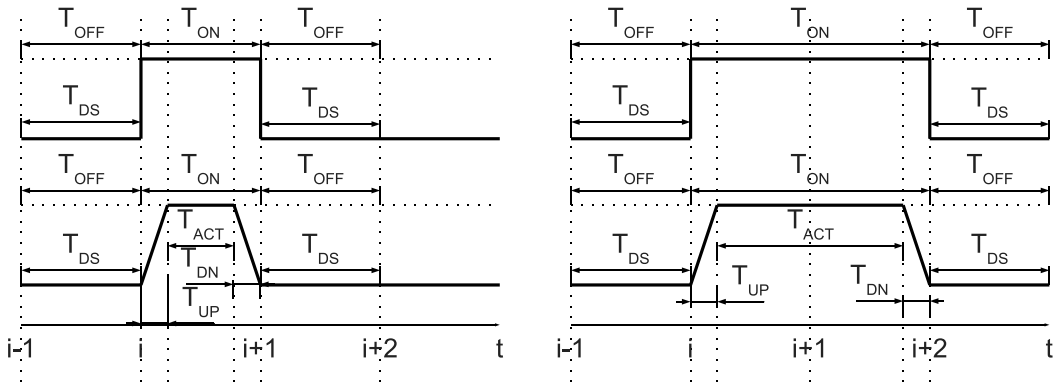
Definition 5 (Connection duration)**.** *Given a pair of sensor nodes, the connection duration ω is the time between the first and last time slot when they are simultaneously and continuously active.*

Clearly, longer time slots bring about longer connection durations, as it is the case for the sample working schedules in Fig. 4.2. The experimental results using Sun SPOT sensors depicted in Fig. 4.3(c) confirm this intuition. For example, for working schedule duty cycle $\mu = 0.05$ the connection duration goes from $\omega = 50$ ms when $T = 50$ ms, to $\omega = 200$ ms when $T = 200$ ms.

4.1.5 Time and Energy Efficiency

Although duty cycling enables energy savings, it is not completely free. In fact, a sensor node faces an overhead cost due to the set up and tear down operations in the transient phases from the deep sleep mode to the normal operation mode, and vice versa. In Fig. 4.4, the ideal and real sensor node behavior are depicted for different active instance lengths. An *active instance* is a set of one or more consecutive time slots during which the sensor node is constantly in the active state. Neither in Fig. 4.4(a), nor in Fig. 4.4(b), does the sensor node execute the application logic throughout T_{ON} as the ideal working schedules on the top of the diagrams state. In fact, an active instance consists of three phases:

1. *Set up,* spanning T_{UP}, during which the sensor node wakes up, updates local data structures (*e.g.*, the alarm for when to switch into the deep sleep mode), (re-)opens network connections, and eventually initializes the sensors;
2. *Activity,* spanning T_{ACT}, during which the sensor node retrieves sensor readings, processes them, and communicates with neighboring nodes; and
3. *Tear down,* spanning T_{DN}, during which the sensor node updates local data structures, closes network connections, eventually resets the sensors, and goes into deep sleep mode[3].



(a) Short active instance

(b) Long active instance

Figure 4.4. Ideal and real duty cycling for (a) short and (b) long active instances. Sensor nodes with longer active instance lengths are more time and energy efficient.

We performed experiments on Sun SPOTs to measure the duration of the set up and tear down phases. The time required by a sensor node to go from deep sleep mode to being ready to execute the application logic depends on the operations during the set up and tear down phases. We decided to maintain a conservative approach and have a minimum set of operations for the sensor to perform during the set up and tear

[3]In studies on energy efficient algorithms [63], tear down costs are usually folded into the set up costs without affecting the validity of the model.

down phases. In our experiments, a sensor node opens a radio connection during the set up phase and closes it during the tear down phase. Even with this very limited set of operations, we found that the set up phase takes around $T_{UP} \simeq 11.50$ ms, while the tear down takes $T_{DN} \simeq 2.86$ ms. In real-life applications, sensors need to perform more complex set up and tear down operations, and thus face an even larger overhead for transitions between deep sleep mode and active state, and vice versa.

Based on the three phases described above, we define the time efficiency for an active instance as follows.

Definition 6 (Time efficiency)**.** *Given an active instance of length T_{ON} with set up, activity, and tear down times T_{UP}, T_{ACT}, and T_{DN}, respectively, its time efficiency is given by*

$$\eta = \frac{T_{ON} - (T_{UP} + T_{DN})}{T_{ON}} = \frac{T_{ACT}}{T_{ON}} \tag{4.1}$$

Energy efficiency is strictly related to the time efficiency as there is an energy cost associated with the set up and tear down operations. Since accurate energy measurements are difficult to perform, in the rest of the chapter we focus on the time efficiency first. Then we compute the energy consumption from the associated time interval and the power consumption in normal operation mode.

We performed additional experiments on Sun SPOTs to measure the time efficiency for different time slot lengths and working schedule duty cycles. From the results shown in Fig. 4.3(d), we observe that the shorter time slots correspond to lower time efficiency. Moreover, for a given time slot length, lower duty cycles yield lower time efficiency. In particular, when the working schedule duty cycle $\mu = 0.05$ and the time slot length $T = 50$ ms, the time efficiency $\eta = 0.83$, which means that the energy associated with 18% of the active time is employed in ancillary operations, not in the application itself. The relationship between duty cycle and time efficiency

can be explained observing that schedules with lower duty cycles feature less active time slots so that there is a lower chance that these active time slots are consecutive and thus form a longer active time instance.

It is easy to observe that $\lim_{T_{ON}\to\infty} \eta = 1$. Thus, longer active time instances bring about higher efficiency. This conclusion would bring the design of the duty cycling scheme in a direction opposite to the one suggested in our analysis of connection delay, when we observed that shorter time slots bring about a performance improvement.

4.1.6 Rationale Behind Markov Chain-based Scheme

We argue that the Markov chain-based solution (see next section) can yield a performance improvement along one metric, while not affecting the other. For instance, we can reduce the time slot length to achieve a lower connection delay, while avoiding a simultaneous reduction in the active instance length. This is insured by tuning the transition probabilities so that the product of the number of consecutive active time slots and the time slot length (*i.e.*, expected active time instance length) is kept constant. In a traditional scheme with *i.i.d.* random variables, this is not feasible because the transition probabilities are uniquely defined by the working schedule duty cycle.

Fig. 4.5 portrays the problem space. In this chapter, we propose to extend the set of metrics by adding a measure of time (energy) efficiency η to the connection delay and duration and aggregate duty cycle already considered in the literature. This equals to turning the 2-dimensional performance space generated by the connection delay δ and connection duration ω dimensions into the 3-dimensional one in the right diagram in Fig. 4.5. In the visual representation, we omit the aggregate duty cycle metric, because it is related only to the working schedule duty cycle, and thus can be considered separately from the other metrics.

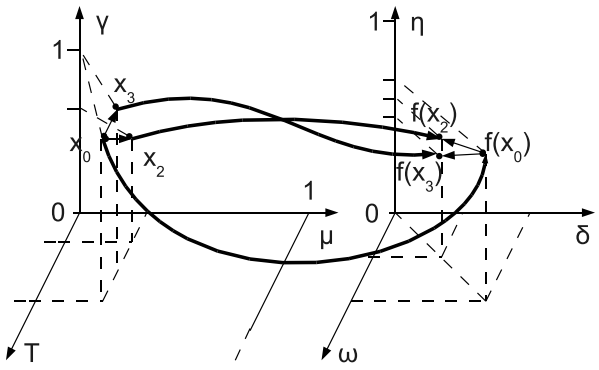


Figure 4.5. The problem space consists of a 3-dimensional control parameter space (left) and a 3-dimensional performance metric space (right). The parameters in the control space are the duty cycle μ, the time slot length T, and the newly added memory coefficient γ. In the performance space, there are the connection delay δ, the connection duration ω, and the proposed new metric for time/energy efficiency η.

Similarly, we turn the 2-dimensional control space defined by the working schedule duty cycle μ and the time slot length T into the 3-dimensional space in the left diagram in Fig. 4.5. The newly added memory coefficient γ represents the amount of memory in the Markov chain employed to generate the randomized working schedules of the wireless sensor nodes.

In Section 4.3, we derive expressions for components $f_\theta(\cdot)$, $f_\delta(\cdot)$, $f_\omega(\cdot)$, and $f_\eta(\cdot)$ of the vector-valued function $\boldsymbol{f} : \mathbb{R}^3 \to \mathbb{R}^4$, where $\boldsymbol{x} = [\mu\ T\ \gamma]'$ is the vector of control parameters and $\boldsymbol{f}(\boldsymbol{x}) = [f_\theta(\cdot)\ f_\delta(\boldsymbol{x})\ f_\omega(\boldsymbol{x})\ f_\eta(\boldsymbol{x})]' = [\theta\ \delta\ \omega\ \eta]'$ is the vector of corresponding performance metrics[4].

Based on the analytical model and the experimental results, we demonstrate how to improve the connection delay δ with limited or no impact on the time and energy efficiency η, moving from $\boldsymbol{f}(\boldsymbol{x_0})$ to $\boldsymbol{f}(\boldsymbol{x_2})$ instead of $\boldsymbol{f}(\boldsymbol{x_3})$ in the performance space of Fig. 4.5. This is achieved by tuning the parameters, including the memory

[4]We use notation v' instead of v^T for transpose of vector v.

coefficient γ, so as to move from $\boldsymbol{x_0}$ in the $\mu \times T$ plane to $\boldsymbol{x_2}$ in the 3-dimensional control space of Fig. 4.5.

4.2 Markov Chain-based Duty Cycling Scheme

Before presenting the Markov chain-based duty cycling scheme, let us first describe our model and assumptions. We assume that sensors are locally time synchronized and feature a common time slot length, T. Although being common across all sensor nodes, the time slot length is not fixed, but is rather computed along with the other input parameters (*i.e.,* working schedule duty cycle and memory coefficient) based on the model in Section 4.3 to achieve optimum performance in terms of connection delay, connection duration, and time and energy efficiency. A time synchronization protocol [64] is periodically executed to correct for clock drifting. The sensors periodically generate working schedules. A working schedule $ws = [s_0, \ldots, s_i, \ldots, s_{N-1}]'$ is an $N \times 1$ binary vector where

$$s_i = \begin{cases} 0 & \text{if sensor is to be dormant at time slot } i, \\ 1 & \text{if sensor is to be active at time slot } i. \end{cases} \tag{4.2}$$

Since the time slot lengths are in the order of hundreds of milliseconds, the time synchronization protocol needs to be performed in the order of tens of minutes to guarantee an accuracy of 2.24 μs, which is more than sufficient for the time slot lengths considered in this scenario. The overhead introduced by FSTP is negligible when the few timestamps (six according to [64]) required to locally synchronize clocks are compared to the period (tens of minutes) for which neighboring sensors are synchronized. Furthermore, given that the time synchronization protocol needs to be performed in the order of tens of minutes while the time slot lengths are in the order of hundreds of milliseconds, the working schedule length N is in the order of 10^4.

The proposed randomized duty cycling scheme is based on a Markov chain as depicted in Fig. 4.6. The Markov chain is defined over the set of states $S = \{0, 1\}$, where $s = 0$ is the dormant state, while $s = 1$ is the active state. A Markov chain is defined by the transition probabilities $\Pr[s_i = 1|s_{i-1} = 0] = \alpha$ and $\Pr[s_i = 0|s_{i-1} = 1] = \beta$. The remaining transition probabilities $\Pr[s_i = 0|s_{i-1} = 0] = 1 - \alpha$ and $\Pr[s_i = 1|s_{i-1} = 1] = 1 - \beta$ can be easily computed based on the values of α and β, thus obtaining the following matrix notation

$$\boldsymbol{P} = [p_{ij}] = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix} \tag{4.3}$$

where $p_{ij} = \Pr[s_t = j|s_{t-1} = i]$. The probability mass function (pmf) of the stationary distribution is

$$\boldsymbol{p} = \begin{bmatrix} p_i \end{bmatrix} = \begin{bmatrix} \mu_0 & \mu_1 \end{bmatrix} = \begin{bmatrix} \frac{\beta}{\alpha+\beta} & \frac{\alpha}{\alpha+\beta} \end{bmatrix} \tag{4.4}$$

where $p_i = \Pr[s = i]$. Thus, $\mu_1 = \Pr[s = 1]$ is the working schedule duty cycle of the wireless sensor nodes[5].

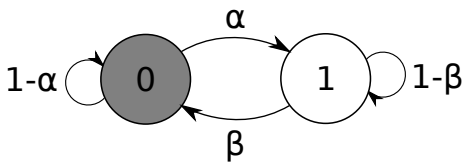


Figure 4.6. Markov chain used to generate working schedules with memory coefficient $\gamma = \alpha + \beta$ and working schedule duty cycle $\mu = \frac{\alpha}{\alpha+\beta}$.

Our scheme is based on the observation that the same value for the duty cycle μ_1 (and thus for the aggregate duty cycle θ, given that θ is related only to μ_1) can be achieved by appropriately scaling the parameters α and β. In fact, $\gamma = \alpha + \beta$ is a

[5]Effectively, both μ and μ_1 represent the duty cycle, although the latter is used in the Markov chain-related derivations for symmetry with μ_0.

α	β	γ	μ
$\frac{1}{20} = 0.05$	$\frac{19}{20} = 0.95$	$\frac{1}{1} = 1.00$	0.05
$\frac{3}{80} = 0.0375$	$\frac{57}{80} = 0.7125$	$\frac{3}{4} = 0.75$	0.05
$\frac{1}{40} = 0.025$	$\frac{19}{40} = 0.475$	$\frac{1}{2} = 0.50$	0.05
$\frac{1}{80} = 0.0125$	$\frac{19}{80} = 0.2375$	$\frac{1}{4} = 0.25$	0.05

Table 4.1. Transition probabilities α and β yielding different memory coefficient γ, but constant duty cycle μ, and thus constant aggregate duty cycle θ.

measure of the memory in the system. As such we refer to γ as the *memory coefficient*. Table 4.1 lists transition probabilities and the corresponding memory coefficient γ and duty cycle μ. While the duty cycle is not affected, the memory coefficient is.

In a WSN scenario, the duty cycle is usually an input parameter to the working schedule generator provided by the energy harvesting controller or set prior to the deployment. The values of α and β can be computed from the target stationary probability μ_1 (*i.e.*, the working schedule duty cycle) and the memory coefficient γ as follows:

$$\begin{cases} \alpha = \gamma\mu_1 \\ \beta = \gamma - \alpha \end{cases} \tag{4.5}$$

Since α and β are probabilities, $0 \leq \alpha, \beta \leq 1$ has to hold, and thus the memory coefficient $\gamma \in \left[0, \frac{1}{1-\mu_1}\right]$. This closed interval can be split in three regions. When $\gamma = 1$, the system is memory-less since the transition probabilities are independent of the state. Thus, using a Markov chain with $\gamma = 1$ to generate a working schedule is equivalent to having *i.i.d.* random variables. For all other values of $\gamma \neq 1$, the system is not memory-less. As γ decreases from 1 to 0, the likelihood of transitions between the states also decreases. When γ increases from 1 to $\frac{1}{1-\mu_1}$, the transitions between the states become more likely.

Our proposed duty cycling scheme exploits the above behavior. Each sensor node generates its random schedule based on a random walk on the Markov chain with transition probabilities α and β computed using the mathematical model derived in the analysis. By generating the working schedule based on a Markov chain with memory coefficient $\gamma < 1$, the time slot length can be reduced and the average active time instance length is kept constant without changing the duty cycle. Shorter time slots yield a lower connection delay, while constant average active time instance length does not negatively affect the energy and time efficiency. The next section presents a mathematical analysis that confirms this intuition, and also provides a set of expressions that can be used for computing the optimal settings of the control parameters.

4.3 Analysis of Markov Chain-based Randomized Scheme

Clearly, the Markov chain-based duty cycling scheme is a generalization of the traditional duty cycling schemes based on *i.i.d.* random variables for each time slot. As a consequence, a larger solution space is available to the designer of duty cycle wireless sensor networks. In order to make the best decision, it is important to have a mathematical model of the proposed scheme. Here we focus our analysis on four performance metrics: aggregate duty cycle, connection delay, connection duration, and time efficiency (energy efficiency is directly related to time efficiency as stated in Section 4.3.5). In this section, we aim to derive expressions for these metrics for any point in the 3-dimensional space defined by time slot length, working schedule duty cycle, and memory coefficient. Given a target energy consumption (*i.e.,* a target aggregate duty cycle), our duty cycling scheme could then be tuned to provide the best performance in terms of time efficiency, connection delay, and connection duration with a given aggregate duty cycle as described below.

4.3.1 Assumptions and Pairwise Markov Chain Model

In the Markov chain of four vertices depicted in Fig. 4.7, each state q represents one of the four combinations of the states of two neighboring wireless sensor nodes:

$$q_i = \begin{cases} 00 & \text{if sensors } a \text{ and } b \text{ are dormant at time slot } i, \\ 01 & \text{if sensor } a \text{ is active and sensor } b \text{ is dormant at time slot } i, \\ 10 & \text{if sensor } a \text{ is dormant and sensor } b \text{ is active at time slot } i, \\ 11 & \text{if sensors } a \text{ and } b \text{ are active at time slot } i. \end{cases} \tag{4.6}$$

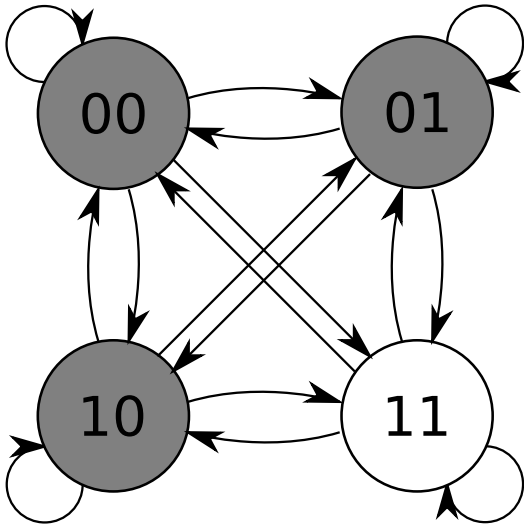


Figure 4.7. Pairwise Markov chain for randomized scheme. A connection is feasible only when both wireless nodes are active (*i.e.,* state $q = 11$).

The transition probabilities between the states are given by the product of the transitions of the wireless sensor nodes (see Fig. 4.6), and thus depend on their working schedule duty cycles μ's and memory coefficients γ's.

$$\boldsymbol{P} = [p_{ij}] = \begin{bmatrix} (1-\alpha)^2 & (1-\alpha)\alpha & \alpha(1-\alpha) & \alpha^2 \\ (1-\alpha)\beta & (1-\alpha)(1-\beta) & \alpha\beta & \alpha(1-\beta) \\ \beta(1-\alpha) & \beta\alpha & (1-\beta)(1-\alpha) & (1-\beta)\alpha \\ \beta^2 & \beta(1-\beta) & (1-\beta)\beta & (1-\beta)^2 \end{bmatrix} \tag{4.7}$$

where $p_{ij} = \Pr[Q_t = j | Q_{t-1} = i]$.

Lemma 4 (Pairwise Markov chain stationary distribution)**.** *The probability mass function (*pmf*) of the stationary distribution for the pairwise Markov chain is*

$$\boldsymbol{p} = \begin{bmatrix} p_i \end{bmatrix} = \begin{bmatrix} \mu_0\mu_0 & \mu_0\mu_1 & \mu_1\mu_0 & \mu_1\mu_1 \end{bmatrix} = \begin{bmatrix} \frac{\beta^2}{(\alpha+\beta)^2} & \frac{\beta\alpha}{(\alpha+\beta)^2} & \frac{\alpha\beta}{(\alpha+\beta)^2} & \frac{\alpha^2}{(\alpha+\beta)^2} \end{bmatrix} \tag{4.8}$$

where $p_i = \Pr[Q = i]$.

(Sketch). The stationary invariant distribution of a Markov chain, whose states are the combination of the states of two independent Markov chains, is the product of the stationary invariant distributions. □

In the next sections, we derive expressions for the performance metrics using the model we introduced above.

4.3.2 Aggregate Duty Cycle

In a Markov chain-based duty cycling scheme, the aggregate duty cycle θ of a wireless node is actually a random variable Θ with expected value

$$E[\Theta] = f_\theta([\mu\ T\ \gamma]') \tag{4.9}$$

where μ, T, and γ are the control parameters in the duty cycling scheme.

Theorem 4 (Expected aggregate duty cycle)**.** *Given a Markov chain-based duty cycling scheme, its expected aggregate duty cycle is given by*

$$E[\Theta] = \mu. \tag{4.10}$$

Proof. Since a wireless node is active only when its working schedule states that it should be active, the expected aggregate duty cycle is equal to the expected value of the state random variable with stationary distribution in (4.4).

$$\begin{aligned} f_\theta([\mu\ T\ \gamma]') &= E[\Theta] \\ &= E[S] \\ &= \mu_1 = \mu. \end{aligned} \tag{4.11}$$

□

As can be observed in Fig. 4.8(a), the experimental results validate the mathematical model for the expected aggregate duty cycle of the Markov chain-based randomized scheme.

4.3.3 Connection Delay

In a Markov chain-based duty cycling scheme, the connection delay δ between a pair of neighboring nodes (say a and b) is actually a random variable Δ with expected value

$$E[\Delta] = f_\delta([\mu\ T\ \gamma]') \tag{4.12}$$

where μ, T, and γ are the control parameters in the duty cycling scheme.

Theorem 5 (Expected connection delay)**.** *Given a Markov chain-based duty cycling scheme, its expected connection delay is given by*

$$E[\Delta] = E[\Delta|Q]\Pr[Q] = T \cdot \boldsymbol{h}'_{opt} \cdot \boldsymbol{p} \tag{4.13}$$

where $\boldsymbol{p}$ *is the stationary invariant distribution of the pairwise Markov chain and* $\boldsymbol{h}_{opt}$ *is the optimal solution to the linear optimization problem*

$$\begin{aligned} \min \quad & \boldsymbol{c} \quad \boldsymbol{h} \\ s.t. \quad & \boldsymbol{A} \quad \boldsymbol{h} = \boldsymbol{b} \\ & \boldsymbol{h} \geq \mathbf{0} \end{aligned} \tag{4.14}$$

with $\boldsymbol{c} = [1\ 1\ 1\ 1]$ *and*

$$\boldsymbol{A} = \begin{bmatrix} -1+(1-\alpha)^2 & (1-\alpha)\alpha & \alpha(1-\alpha) & \alpha^2 \\ (1-\alpha)\beta & -1+(1-\alpha)(1-\beta) & \alpha\beta & \alpha(1-\beta) \\ \beta(1-\alpha) & \beta\alpha & -1+(1-\beta)(1-\alpha) & (1-\beta)\alpha \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{4.15}$$

$$\boldsymbol{b} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 0 \end{bmatrix} \tag{4.16}$$

Proof. By simple manipulations of the expected connection delay, we obtain

$$\begin{aligned} E[\Delta] &= \sum_q E[\Delta|Q=q]\Pr[Q=q] \\ &= T \cdot \sum_q E[H|Q=q]\Pr[Q=q]. \end{aligned} \tag{4.17}$$

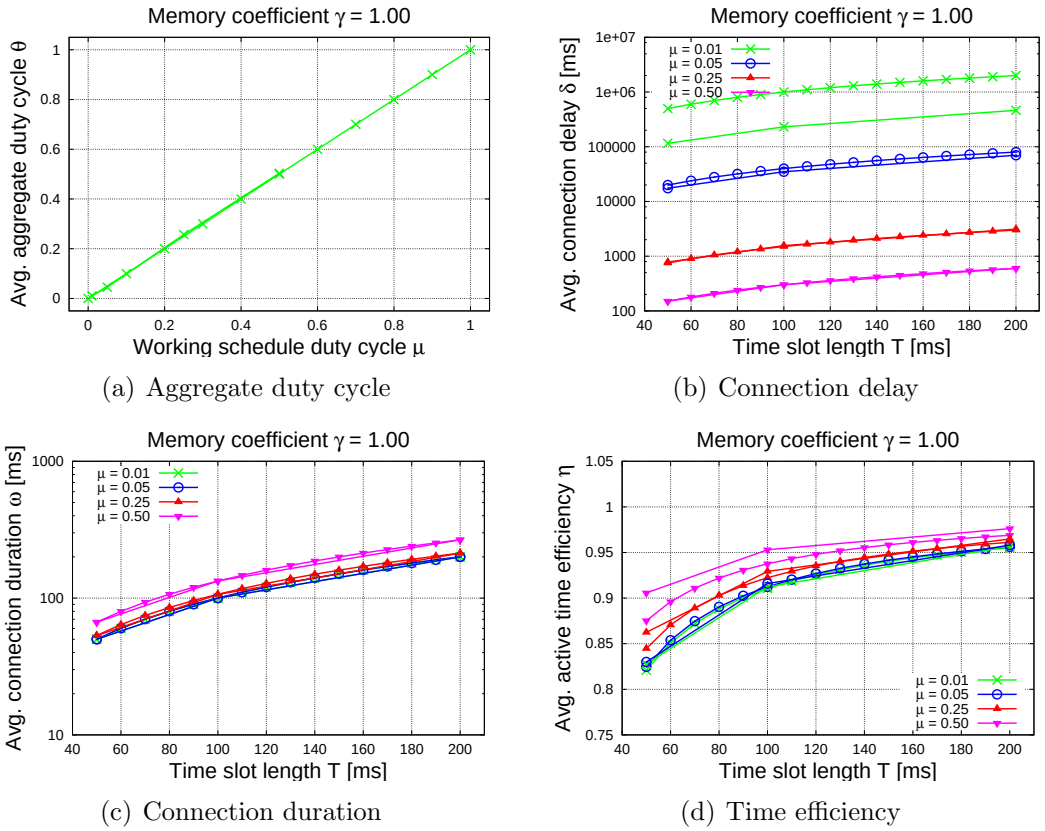
where $E[H|Q]$ is the expected number of time slots until both sensors are active, given the current state Q. As such, $E[H|Q]$ are the expected hitting times of the state $q = 11$ in the pairwise Markov chain in Fig. 4.7. The expected hitting times

$\boldsymbol{h_{opt}} = E[H|Q]$ can be computed by finding the minimal non-negative solution to the system of linear equations

$$\begin{cases} E[H|Q=00] = 1 & +E[H|Q=00]\Pr[Q=00|Q=00]+ \\ & +E[H|Q=01]\Pr[Q=00|Q=01]+ \\ & +E[H|Q=10]\Pr[Q=10|Q=00]+ \\ & +E[H|Q=11]\Pr[Q=11|Q=00] \\ E[H|Q=01] = 1 & +E[H|Q=00]\Pr[Q=00|Q=01]+ \\ & +E[H|Q=01]\Pr[Q=01|Q=01]+ \\ & +E[H|Q=10]\Pr[Q=10|Q=01]+ \\ & +E[H|Q=11]\Pr[Q=11|Q=01] \\ E[H|Q=10] = 1 & +E[H|Q=00]\Pr[Q=00|Q=10]+ \\ & +E[H|Q=01]\Pr[Q=01|Q=10]+ \\ & +E[H|Q=10]\Pr[Q=10|Q=10]+ \\ & +E[H|Q=11]\Pr[Q=11|Q=10] \\ E[H|Q=11] = 0 & \end{cases} \tag{4.18}$$

which is equivalent to solving the linear optimization problem in (4.14). □

As can be observed in Fig. 4.8(b), the mathematical model of (4.13) precisely matches the experimental results. The discrepancy for $\mu = 0.01$ is due to the small sample size for that specific experiment.



(a) Aggregate duty cycle

(b) Connection delay

(c) Connection duration

(d) Time efficiency

Figure 4.8. Analytical (0.10-spaced) and experimental ($\mu = 0.01, 0.05, 0.25, 0.50$) results for (a) aggregate duty cycle vs. working schedule duty cycle; and analytical (10 ms-spaced) and experimental ($T = 50, 100, 200$) results for (b) connection delay, (c) connection duration, and (d) time efficiency vs. time slot length for different duty cycles when the working schedule is generated with independently and identically distributed (*i.i.d.*) random variables.

4.3.4 Connection Duration

Similar to the connection delay, the connection duration ω between a pair of sensor nodes is also a random variable Ω and its expected value $E[\Omega] = f_\omega([\mu\ T\ \gamma]')$ can be derived for a Markov chain-based duty cycling scheme.

Theorem 6 (Expected connection duration)**.** *Given a Markov chain-based duty cycling scheme, its expected connection duration is given by*

$$E[\Omega] = \frac{T}{1-(1-\beta)^2}. \tag{4.19}$$

Proof. The *pmf* of the random variable V representing the number of consecutive transitions on the self loop of state $q = 11$ in Fig. 4.7 (*i.e.*, the number of time slots in the connection) is a geometric distribution with success probability $p = 1-(1-\beta)^2$.

$$\begin{aligned} f_\omega([\mu\ T\ \gamma]') &= E[\Omega] \\ &= T \cdot E[V] \\ &= T \cdot \sum_{v=1}^{\infty} \left[(1-\beta)^{2(v-1)}(1-(1-\beta)^2)\right] \\ &= \frac{T}{1-(1-\beta)^2} \end{aligned} \tag{4.20}$$

□

Given that $E[\Omega]$ is only a function of β (and T) and not α, it is possible to tune the expected connection duration without affecting the duty cycle. The time slot length T has no effect on the duty cycle because it does not affect the ratio of active and dormant time slots. In fact, Fig. 4.8(c) shows that the analytical model of (4.19) matches the experimental results for the average connection duration.

4.3.5 Time Efficiency

We analyze the time efficiency of working schedules generated by our proposed randomized duty cycling scheme and derive a formula to compute the expected time efficiency in terms of the working schedule duty cycle, memory coefficient, and time slot length. The time efficiency η is actually a random variable because the duty cycling scheme has no deterministic control on the length of active time instances.

Theorem 7 (Expected time efficiency)**.** *Given a Markov chain-based duty cycling scheme, its expected time efficiency is derived as*

$$E[\eta(K)] = 1 + \frac{T_{UP} + T_{DN}}{T} \cdot \frac{\beta \ln(\beta)}{1-\beta}. \tag{4.21}$$

Proof. Based on Definition 6, the time efficiency $\eta(k)$ of an active instance consisting of $k \geq 1$ consecutive time slots is given by

$$\eta(k) = \frac{kT - (T_{UP} + T_{DN})}{kT} = 1 - \frac{T_{UP} + T_{DN}}{kT}, \tag{4.22}$$

where T is the time slot length, T_{UP} and T_{DN} are respectively the time costs for the set up and tear down operations.

The number k of consecutive time slots in an active instance is a random variable K with geometric distribution, and thus its *pmf* is

$$\Pr[K = k] = (1 - \beta)^{k-1}\beta, \tag{4.23}$$

where $\beta = \Pr[s_i = 0 | s_{i-1} = 1]$ and $1 - \beta = \Pr[s_i = 1 | s_{i-1} = 1]$ and β is computed from the working schedule duty cycle μ_1 and memory coefficient γ as in (4.5).

Given the formula for time efficiency in (4.22) and the *pmf* of the number K of time slots in an active instance in (4.23), the expected time efficiency $E[\eta(K)]$ can be computed as

$$\begin{aligned}
E[\eta(K)] &= \sum_{k=1}^{\infty} \eta(k) \Pr[K = k] \\
&= \sum_{k=1}^{\infty} \left(1 - \frac{T_{UP} + T_{DN}}{kT}\right) (1 - \beta)^{k-1}\beta \\
&= \beta \sum_{k=1}^{\infty} \left[(1 - \beta)^{k-1}\right] - \beta \frac{T_{UP} + T_{DN}}{T} \sum_{k=1}^{\infty} \left[\frac{(1 - \beta)^{k-1}}{k}\right] \\
&= 1 - \beta \frac{T_{UP} + T_{DN}}{T} \sum_{k=1}^{\infty} \frac{(1 - \beta)^{k-1}}{k} \\
&= 1 - \beta \frac{T_{UP} + T_{DN}}{T} \left(-\frac{\ln(\beta)}{1 - \beta}\right) \\
&= 1 + \frac{T_{UP} + T_{DN}}{T} \cdot \frac{\beta \ln(\beta)}{1 - \beta},
\end{aligned} \tag{4.24}$$

where the solution to the last sum is based on the Taylor series expansion of the natural logarithm $\ln(1 - p) = -\sum_{k=1}^{\infty} \frac{p^k}{k}$. □

Both the experimental and analytical results for the time efficiency are plotted in Fig. 4.8(d). The analytical values computed according to (4.21) are lower than the experimental ones. This difference can be explained by the fact that we use a constant value for the set up and tear down costs T_{UP} and T_{DN}, whereas in the experiments these are random variables. Therefore, the occurrences of values lower than the average have a stronger impact on the time efficiency than occurrences of values greater than the average.

Even though we were able to derive a closed-form expression for the expected time efficiency, it is worthwhile analyzing two bounds as they provide further insight.

Theorem 8 (Lower bound on expected time efficiency)**.** *Given a Markov chain-based duty cycling scheme, the lower bound on the expected time efficiency is given by*

$$E[\eta(K)] > \eta(1). \tag{4.25}$$

(Sketch). Apply the inequality $\sum_{k=1}^{\infty} \frac{(1-\beta)^{k-1}}{k} < \sum_{k=1}^{\infty}(1-\beta)^{k-1} = \frac{1}{\beta}$ to (4.24). □

The result in (4.25) is intuitive as it states that the expected time efficiency of the randomized duty cycling scheme is always higher than the one of an active instance consisting of only one time slot.

Theorem 9 (Upper bound on expected time efficiency)**.** *Given a Markov chain-based duty cycling scheme, the upper bound on the expected time efficiency is*

$$E[\eta(K)] < \eta\left(\frac{1}{\beta}\right). \tag{4.26}$$

Proof. A corollary of Jensen's inequality states that a concave function of the expected value is greater than the expected value of the concave function:

$$E[f(X)] < f(E[X]) \tag{4.27}$$

Given that $\eta(\cdot)$ in (4.22) is a concave function, we obtain

$$E[\eta(K)] < \eta\left(\frac{1}{\beta}\right). \tag{4.28}$$

□

Equation (4.26) states that the expected time efficiency is always lower than the time efficiency of an active instance that consists of the expected number of time slots for a given transition probability β. By putting together the two bounds, the expected time efficiency is

$$\eta(1) < E[\eta(K)] < \eta\left(\frac{1}{\beta}\right) \tag{4.29}$$

A tighter upper bound on the expected time efficiency can be computed based on the observation that the length k of any active time instance is upper bounded by the finite length N of the working schedule. Therefore, we can compute a weighted version of the *pmf* such that $\sum_{k=1}^{N} \Pr[K = k] = 1$, and solve (4.24) numerically.

Energy efficiency is tightly related to time efficiency. If the power consumption is assumed to be constant during normal operation mode, time and energy efficiency are equal, and thus the results on time efficiency can be directly extended to energy efficiency and consumption.

In the next section, we present experimental results that support the analysis of connection delay, duration, and time efficiency in our Markov chain-based duty cycling scheme.

4.4 Experimental Results for Randomized Scheme

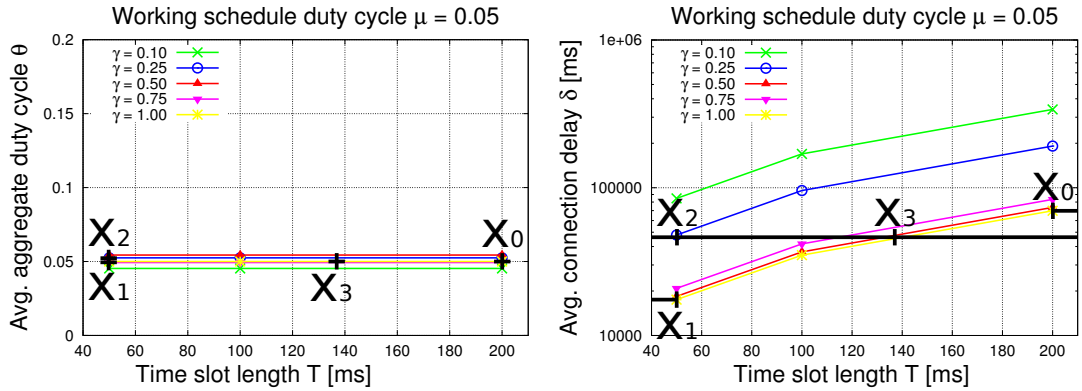
In order to validate the proposed Markov chain-based duty cycling scheme and the mathematical analysis, we performed a series of experiments with wireless sensor nodes, and measured the connection delay, connection duration, time efficiency, and aggregate duty cycle. We employed Sun SPOT sensors for our experiments, although any other sensor platform could be used for experimentation. Sun SPOTs implement IEEE 802.15.4 PHY and MAC standards, and use the Link Quality Routing Protocol

(LQRP) at the network layer [65]. It is worthwhile observing that different platforms eventually yield different results as platform-specific hardware affects the cost – both in terms of time and associated energy – of set up and tear down operations.

Not only does the hardware, but also the software has an impact on the costs when switching between dormant and active states. In fact, different protocols at the link, network, transport, and application layers might require operations that bring about very different costs. Keeping a conservative approach in our experiments, Sun SPOTs simply open and close a radiogram connection during the set up and tear down phases, respectively. These operations represent the minimum required for communicating with neighboring nodes. Clearly, performing additional operations such as neighbor discovery would add to the set up and tear down costs, thus furthering our claim for control on time and energy efficiency.

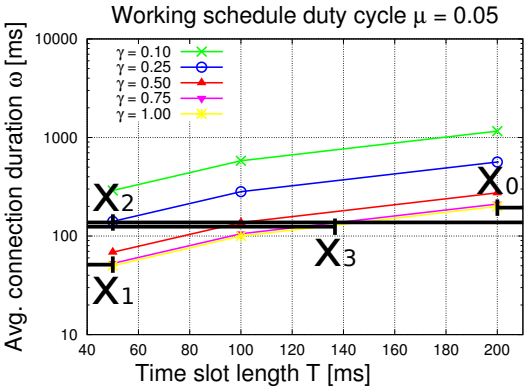
We performed experiments for four different working schedule duty cycles ($\mu = 0.01, 0.05, 0.25, 0.50$), five memory coefficients ($\gamma = 0.10, 0.25, 0.50, 0.75, 1.00$), and three time slot lengths ($T = 50, 100, 200$ ms). These values cover the range from medium to very low duty cycle, the interval between 0.10 and 1 for the memory coefficient, and time slot length with order of magnitude equal to or greater than the one for set up and tear down operations ($T_{UP} + T_{DN} = 14.36$ ms). In all experiments, the working schedules consist of $6,000$ time slots. According to the mathematical analysis in the previous section, in the experiments we consider pairs of wireless sensor nodes, and compute the one-hop connection delay and duration, and the time efficiency of the sensor nodes.

Fig. 4.9 displays results for connection delay, connection duration, and time efficiency vs. the time slot length for all memory coefficients and duty cycle $\mu = 0.05$. In our discussion on the experiments, we focus on the results for points $\boldsymbol{x_0}$ through

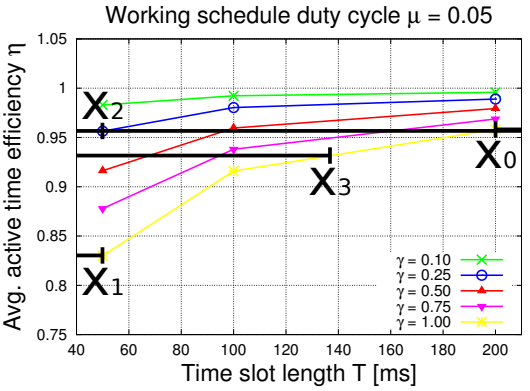


(a) Aggregated duty cycle, $\mu = 5\%$

(b) Connection delay, $\mu = 5\%$



(c) Connection duration, $\mu = 5\%$



(d) Time efficiency, $\mu = 5\%$

Figure 4.9. (a) Aggregate duty cycle, (b) connection delay, (c) connection duration and (d) time efficiency vs. time slot length for variable memory coefficient $\gamma = \alpha + \beta$ and duty cycle $\mu = 0.05$ for Sun SPOTs. By combining a reduction of the time slot length with a lower memory coefficient, the connection delay can be reduced, while not degrading the time efficiency. Similarly, the connection duration and time efficiency can be improved without affecting the connection delay, if the increase in the time slot length is accompanied by a higher memory coefficient.

$\boldsymbol{x_3}$ listed in Table 4.2 and marked in Fig. 4.9 for wireless sensor nodes with duty cycle $\mu = 0.05$, although similar conclusions can be drawn for other duty cycles as well.

First of all we observe that all solutions $\boldsymbol{x_0}$ yield equal aggregate duty cycle θ, and thus feature equal energy consumption. Therefore, we can perform a fair comparison of the solutions in terms of the other performance metrics, including connection delay and time efficiency.

Point	Memory coeff. γ	Time slot length T [ms]	Time eff. η (% diff.)	Conn. delay δ [ms] (% diff.)	Conn. dur. ω [ms] (% diff.)
$\boldsymbol{x_0}$	1.00	200	0.9581 ($\pm$0.00%)	69,896 ($\pm$0.00%)	200 ($\pm$0.00%)
$\boldsymbol{x_1}$	1.00	50	0.8300 ($-$13.37%)	17,474 ($-$75.00%)	50 ($-$75.00%)
$\boldsymbol{x_2}$	0.25	50	0.9563 ($-$0.19%)	47,854 ($-$31.54%)	141 ($-$29.50%)
$\boldsymbol{x_3}$	1.00	137	0.9315 ($-$2.78%)	47,854 ($-$31.54%)	137 ($-$31.50%)

Table 4.2. Time efficiency, connection delay, and connection duration for Sun SPOTs with duty cycle $\mu = 0.05$ and $T_{UP} = 11.50\ ms$ and $T_{DN} = 2.86$ ms.

Given the setting with time slot length $T = 200$ ms and memory coefficient $\gamma = 1.00$ ($\boldsymbol{x_0}$), the initial duty cycling scheme has an average connection delay $\delta = 69,896$ ms with time efficiency $\eta = 0.9581$. When the time slot length is reduced from $T = 200$ ms to $T = 50$ ms and the memory coefficient $\gamma = 1.00$ is upheld ($\boldsymbol{x_1}$), the duty cycling metrics are $\delta = 17,474$ ms and $\eta = 0.8300$. Besides featuring a lower connection delay, this setting also yields a lower time efficiency, which represents a negative side effect of the time slot length reduction.

Now, adopting our Markov chain-based duty cycling scheme with time slot length $T = 50$ ms and memory coefficient $\gamma = 0.25$ ($\boldsymbol{x_2}$) brings about a connection delay $\delta = 47,854$ ms, which represents a 31.54% reduction on the original value ($\boldsymbol{x_0}$), while keeping a time efficiency $\eta = 0.9563$. Although the memory-less setting with $T = 137$ ms and $\gamma = 1.00$ ($\boldsymbol{x_3}$) achieves the same connection delay $\delta = 47,854$ ms, it yields a time efficiency $\eta = 0.9315$, which is 2.59% below the 0.9563 efficiency of the Markov chain-based duty cycling scheme. It is easy to observe from Table 4.2 that

the results for the connection duration are directly related to the connection delay, given that all settings have duty cycle $\mu = 0.05$.

Given a power consumption between 210-360 mW for a Sun SPOT [66], the Markov chain-based duty cycling scheme can reduce the energy consumption by 312-536 mJ per minute.

In the next chapter, we extend the Markov chain-based solution to a partially randomized duty cycling schemes [59], where a wireless node is active not only when asserted by its working schedules, but also when it has outgoing messages and the next-hop neighbor is active.

4.5 Related Work

Over the last decade there has been a large amount of work on duty cycling in wireless sensor networks. Research has focused especially on the design of schemes that optimize performance in terms of sensing (*e.g.*, coverage of an area of interest) and communication (*e.g.*, data forwarding delay), given a duty cycle.

Research on connectivity and coverage often relies on results from random (geometric) graph theory [67, 68]. A duty cycle WSN can be modeled as a random (geometric) graph, since neighboring nodes have a non-zero probability p of being logically connected, which depends on the duty cycles. Furthermore, coverage and connectivity are monotone properties of random graphs. That is, a random graph with a higher edge probability p, either due to higher duty cycle, or a higher node density, is at least as likely to cover an area or be connected. Results for thresholds of monotone properties (*e.g.*, [69]), are useful for designing randomized duty cycling schemes that improve connectivity and reduce delay.

In [70] and [71], algorithms have been presented to provide area coverage in duty cycle WSNs. Although the notion of sensor set up time was mentioned in [71], it was not really taken into account in the algorithm design.

As far as communication is concerned, there exist works in the literature that focus on duty cycle at the medium access control (MAC) layer. For example, S-MAC [17] was proposed to minimize energy consumption in battery-powered wireless sensor nodes. B-MAC [18] aims to reduce costs due to synchronization in S-MAC and other protocols such as T-MAC [72] by means of long preambles and low power listening. SCP-MAC [73] is a hybrid solution between S-MAC and B-MAC, which relies on scheduled channel pollings instead of asynchronous preambles. Instead, X-MAC [20] improves on B-MAC by means of short and strobed preambles which reduce the latency due to long preambles in B-MAC.

More recently, a cross-layer approach in R-MAC [74] aims to reduce energy latency introduced by S-MAC. In [75], a solution is proposed to reduce end-to-end delay in case of anycast at the MAC layer. Additionally, an algorithm is proposed in [76] to optimize the expected data delivery ratio and communication delay or energy consumption based on a forwarding set, instead of a single node for the next hop. A different solution to B-MAC's long preambles is proposed in RI-MAC [77], a receiver-initiated MAC where the sender waits for a beacon from the receiver before forwarding the packet. Many of the solutions in these MAC protocols are evaluated in a systematic way and organized in a comprehensive architecture in A-MAC [78].

Other works address specific communication operations, including data dissemination and collection without specifically focusing on the MAC layer. In case of data dissemination, the energy-optimal tree is expanded in [59] to reduce latency and improve reliability based on forwarding sets, while in [79] a distributed dynamic programming algorithm is designed for this goal. In the graph theoretic analysis of

duty cycling in WSNs [80], it is shown that the connection delay is reduced if the number of time slots is increased, while the duty cycle is fixed. More recent work [81] considers duty cycling with respect to communication in an energy harvesting WSN. In this scenario, the amount of available energy is space- and time-dependent. Hence, a flexible duty cycling solution is developed that improves communication latency using the currently available energy supply.

To the best of our knowledge, ours is the first work that considers the energy and time efficiency decay incurred by duty cycling due to set up and tear down costs and provides a solution to improve efficiency, while keeping a constant connection delay, or improve connection delay yet not negatively affecting efficiency. Thus, our proposed solution is orthogonal to related work in the area of duty cycling in wireless sensor networks. A direct comparison between the proposed schemes and existing protocols is not required because concepts from both can be incorporated in a more advanced duty cycling protocol to achieve an improvement on the overall energy efficiency beyond what either scheme can offer.

4.6 Summary

In this chapter, we introduced a metric for time and energy efficiency of duty cycling in WSNs, and presented a Markov chain-based, randomized scheme. We analyzed the proposed scheme to derive mathematical expressions for the aggregate duty cycle, connection delay and duration as well as time and energy efficiency, given the duty cycle, time slot length, and memory coefficient of the Markov chain.

Experimental results on real hardware validate the performance improvements achieved by the proposed scheme, as opposed to memory-less schemes. In fact, in case of Sun SPOT-based WSNs with minimal set up and tear down operations, the

Markov chain-based scheme yields a 2.59% increase in time efficiency, which saves up to 536 mJ per minute per wireless node.

CHAPTER 5

MARKOV-CHAIN BASED PARTIALLY RANDOMIZED DUTY CYCLING SCHEMES

In this chapter, we apply the know-how to design an energy-efficient partially randomized scheme, where wireless nodes share working schedules so that they can switch into active state whenever they have packets to relay and the next-hop neighbor is active[1]. We derive expressions for the connection delay, connection duration, and time and energy efficiency, as well as the aggregate duty cycle.

We validate our analysis with experiments on Sun SPOTs, showing that the expected connection delay can be reduced by at least 49.12%, while keeping a time efficiency of 0.9248, as opposed to a 0.44% reduction in a scheme using *i.i.d.* random variables. In turn, this corresponds to a saving between 51-89 mJ per minute per wireless node.

Finally, we use the analytical models to compare the two proposed Markov chain-based schemes and show how a partially randomized scheme outperforms a randomized one in terms of connection delay and duration with constant energy consumption and efficiency, while facing a small additional cost for distributing working schedules among neighboring wireless nodes.

In Section 5.1 we introduce the Markov chain-based partially randomized scheme and we analyze it in Section 5.2. The experimental results are presented in Section

[1]The research work presented in this chapter has been published in the Proc. of the 31st IEEE International Conference on Distributed Computing Systems (ICDCS 2011) [60], and the ACM Journal on Emerging Technologies in Computing (JETC) [61].

5.3. In Section 5.4 we compare the two randomized schemes, and we discuss optimal duty cycling design in Section 5.5.

5.1 Partially Randomized Markov Chain-based Duty Cycling Schemes

A partially randomized scheme, such as the one introduced in [59], is an extension of the randomized scheme analyzed in the previous chapter, as it is apparent when comparing the block diagrams in Figures 5.1 and 4.1. The objective of a partially randomized scheme is to use the information about the neighbors' states to improve the performance, in terms of end-to-end delay, by eventually switching into the active state when the next-hop node is active. Therefore, wireless nodes share the generated working schedules with their neighbors. Clearly, the more wireless nodes a working schedule is distributed to, the larger performance improvement in terms of end-to-end delay can be achieved, given that routes with shorter delays can be selected. Nonetheless, the working schedule distribution process costs in terms of time and energy and the marginal performance improvement rapidly decays as a function of the distance between sensors. Therefore, the working schedule distribution should be limited to wireless nodes within a few (one or two) hops.

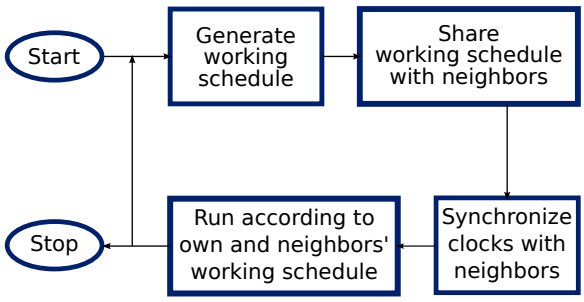


Figure 5.1. Block diagram for partially randomized scheme operation.

In fact, a more efficient solution to distribute the working schedules can be thought of. Instead of distributing the working schedules, we propose that *(i)* wireless nodes distribute only the *seed* used by the pseudorandom number generator to create their schedules, and *(ii)* their neighbors locally re-compute the working schedules. Given that transmitting a bit over the radio is more energy expensive than computing it on a microcontroller, this solution is expected to reduce the energy cost associated with the working schedule distribution. Furthermore, the time synchronization messages can be exploited to carry the seeds as their payload, thus reducing the time required for initialization.

In the next section, we present our mathematical analysis of the performance of the Markov chain-based partially randomized scheme.

5.2 Analysis of Partially Randomized Duty Cycling Scheme

In our analysis of the proposed Markov chain-based partially randomized scheme, we derive mathematical formulas for the connection delay, connection duration, time efficiency, and aggregate duty cycle, given the time slot length, working schedule duty cycle and memory coefficient. This mathematical model can then be employed to set the control parameters in order to achieve a given performance. Before presenting our analytical results, we introduce the model and assumptions.

5.2.1 Assumptions and Pairwise Markov Chain Model

Unlike the case of the randomized scheme in Section 4.2, the aggregate duty cycle θ of a partially randomized scheme is not necessary equal to the working schedule duty cycle μ. In fact, the aggregate duty cycle depends not only on the wireless node's duty cycle, but also on *(i)* the neighbors' schedules and *(ii)* the amount of data that the wireless node is transmitting or relaying.

The aggregate duty cycle depends on the neighbors' schedules, because a wireless node can switch to active state also when not stated by its working schedule in order to forward a message to a neighbor. Furthermore, the aggregate duty cycle depends on the number of one-hop neighbors because the more neighbors a wireless node has, the more likely it is to switch into active state to forward a message to one of them. However, a wireless node does not have to be active all the time slots when its neighbors are, but rather only when it has data to relay to one of them. Thus, in a partially randomized scheme the aggregate duty cycle does not depend only on network features such as topology and schedules, but also on application-specific features such as the amount of traffic.

In our analysis of the performance of the proposed Markov chain-based partially randomized scheme, we assume that *(i)* a wireless node relays data to only one neighbor; and *(ii)* the transmit buffer is never empty. We argue that this case is representative of many sensor network applications where wireless nodes relay sensed data along a routing tree rooted at the base station. The analysis of the performance for other operational modes (*e.g.,* one-to-many in data dissemination) is left for future work.

The Markov chain model for a pair of neighboring wireless nodes is similar to the one in Section 4.3.1 used in the analysis of the randomized scheme. The transition matrix and the stationary distribution are equal to the ones defined in (4.7) and (4.8), respectively.

However, given the assumptions on the wireless sensor network, the joint state of a pair of wireless nodes depicted in Fig. 5.2 is different from the one of Fig. 4.7 for the randomized scheme. In fact, in the diagram of Fig. 5.2 wireless node a is in the active state not only when it is stated so by its working schedule (*i.e.,* $q \in \{01, 11\}$),

but also whenever its neighbor b is in the active state according to its schedule (*i.e.,* $q = 10$).

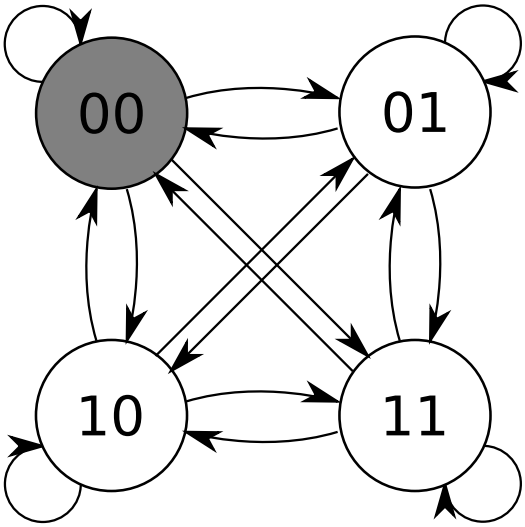


Figure 5.2. Pairwise Markov chain model for partially randomized scheme. A wireless node is active not only when it is stated by its working schedule (*i.e.,* state $q \in \{01, 11\}$), but also when the receiving wireless node is active (*i.e.,* state $q = 10$).

In the following sections, we present the mathematical analysis for aggregate duty cycle, connection delay and duration, and time efficiency based on the pairwise Markov chain model that we introduced.

5.2.2 Aggregate Duty Cycle

The aggregate duty cycle θ of a partially randomized Markov chain-based scheme is actually a random variable Θ. Based on the assumptions and the model in the previous section, we derive the following expression for its expected value.

Theorem 10 (Expected aggregate duty cycle)**.** *The expected aggregate duty cycle of a partially randomized scheme is*

$$E[\Theta] = 2 \cdot \mu - \mu^2 \tag{5.1}$$

Proof. The aggregate duty cycle of a wireless node a can be computed adding the working schedule duty cycle of the wireless node b to that of node a. The expected ratio of time during which they are independently both active should be subtracted because it has been counted twice (*i.e.,* once for a and once for b). Thus,

$$\begin{aligned} E[\Theta] &= E[S_a] + E[S_b] - E[S_a, S_b] \\ &= \mu + \mu - \mu \cdot \mu \\ &= 2 \cdot \mu - \mu^2. \end{aligned} \tag{5.2}$$

where $E[S_a, S_b] = E[S_a] \cdot E[S_b]$ because working schedules on different wireless nodes are generated independently. □

We performed experiments for the proposed Markov chain-based partially randomized scheme on Sun SPOT sensors. The experimental results plotted in Fig. 5.3(a) validate the analytical model. Furthermore, the plot of the analytical results confirm that (5.1) yields an expected aggregate duty cycle $E[\Theta] \in [0, 1]$, $\forall \mu \in [0, 1]$.

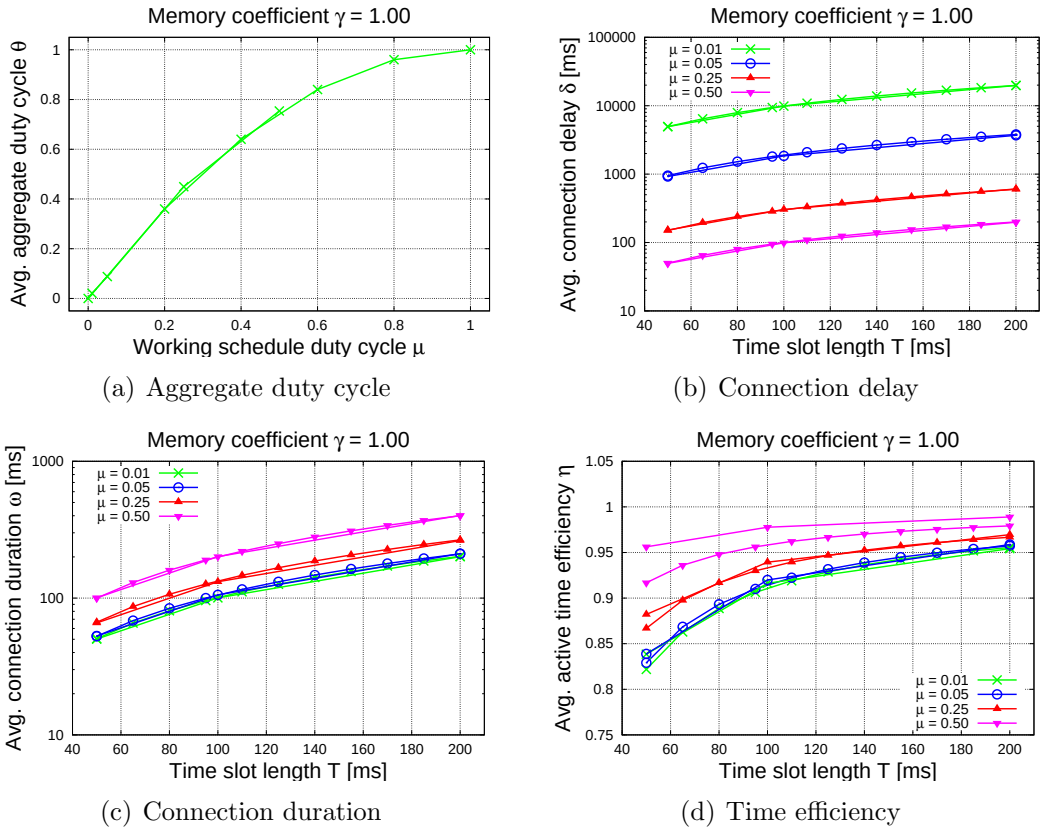
It is also worthwhile observing that the aggregate duty cycle θ features the highest increase with respect to the working schedule duty cycle μ when $\mu \to 0$, since

$$\frac{d}{d\mu}(2\mu - \mu^2) = 2 - 2\mu. \tag{5.3}$$

This is intuitive as wireless nodes are less likely to be independently both active, when they feature a lower duty cycle.

5.2.3 Connection Delay

In a partially randomized Markov chain-based scheme, the connection delay δ between a pair of neighboring nodes is a random variable Δ.



(a) Aggregate duty cycle

(b) Connection delay

(c) Connection duration

(d) Time efficiency

Figure 5.3. Analytical (0.10-spaced) and experimental ($\mu = 0.01, 0.05, 0.25, 0.50$) results for (a) aggregate duty cycle, and analytical (10 ms-spaced) and experimental ($T = 50, 100, 200$) results for (b) connection delay, (c) connection duration, and (d) time efficiency vs. time slot length for different duty cycles when the working schedule is generated with independently and identically distributed (*i.i.d.*) random variables in a partially randomized scheme.

Theorem 11 (Expected connection delay)**.** *Given a partially randomized Markov chain-based duty cycling scheme, its expected connection delay is given by*

$$f_\delta([\mu\ T\ \gamma]') = E[\Delta] = \frac{\beta}{\alpha(\alpha+\beta)} \tag{5.4}$$

Proof. Since the transmitting node a in a partially randomized scheme can switch into active state whenever the receiving node b does, the connection delay depends only on the state S_b of the receiving node b. In fact, the number of time slots until a connection is established is the random variable U representing the number of

consecutive transitions on the self loop of state $S_b = 0$ in Fig. 4.6. Given that U has a geometric distribution with success probability α the expected connection delay is given by

$$\begin{aligned} f_\delta([\mu\ T\ \gamma]') &= E[\Delta] \\ &= \sum_s E[\Delta|S_b = s] \Pr[S_b = s] \\ &= T \cdot E[U|S_b = 0] \Pr[S_b = 0] \\ &= T \cdot \sum_{u=1}^{\infty} \left[(1-\alpha)^{u-1}\alpha\right] \cdot \frac{\beta}{\alpha+\beta} \\ &= T \cdot \frac{\beta}{\alpha(\alpha+\beta)} \end{aligned} \tag{5.5}$$

□

The results of experiments on Sun SPOT sensors for the connection delay plotted in Fig. 5.3(b) validate the analytical model in (5.4).

5.2.4 Connection Duration

In a partially randomized Markov chain-based scheme, the connection duration ω between a pair of neighboring nodes is a random variable Ω.

Theorem 12 (Expected connection duration)**.** *Given a partially randomized Markov chain-based duty cycling scheme, its expected connection duration is given by*

$$f_\omega([\mu\ T\ \gamma]') = E[\Omega] = \frac{T}{\beta}. \tag{5.6}$$

Proof. Since the transmitting node a in a partially randomized scheme can be active at any time slot when the receiving node b is active, the connection duration depends only on the state S_b of the receiving node b. In fact, the number of time slots in the connection is the random variable V representing the number of consecutive

transitions on the self loop of state $S_b = 1$ in Fig. 4.6. Given that V has a geometric distribution with success probability β the expected connection duration is given by

$$\begin{aligned} f_\omega([\mu\ T\ \gamma]') &= E[\Omega] \\ &= T \cdot E[V] \\ &= T \cdot \sum_{v=1}^{\infty} \left[(1-\beta)^{v-1}\beta\right] \\ &= \frac{T}{\beta} \end{aligned} \tag{5.7}$$

□

The analytical model for the connection duration in (5.6) is confirmed by the experimental results for Sun SPOTs plotted in Fig. 5.3(c).

5.2.5 Time Efficiency

In a partially randomized Markov chain-based scheme, the time efficiency η for a wireless node is a random variable Ω.

Theorem 13 (Expected time efficiency)**.** *Given a partially randomized Markov chain-based duty cycling scheme, its expected time efficiency is derived as*

$$f_\eta([\mu\ T\ \gamma]') = E[\eta(K)] = 1 + \frac{T_{UP} + T_{DN}}{T} \cdot \frac{\frac{[1-(1-\alpha)^2]\beta}{(\alpha+\beta)^2-\beta^2} \ln\left(\frac{[1-(1-\alpha)^2]\beta}{(\alpha+\beta)^2-\beta^2}\right)}{1 - \left[\frac{[1-(1-\alpha)^2]\beta}{(\alpha+\beta)^2-\beta^2}\right]}. \tag{5.8}$$

Proof. Given the 2-state Markov chain in Fig. 5.4 that is equivalent, as far as transitions between states $q = 00$ and $q \in \{01, 11, 10\}$ are concerned, to the 4-state Markov chain in Fig. 5.2, the time efficiency of the partially randomized scheme is given by

$$E[\eta(K)] = 1 + \frac{T_{UP} + T_{DN}}{T} \cdot \frac{\beta' \ln(\beta')}{1 - \beta'} \tag{5.9}$$

by substituting β' for β in (4.21).

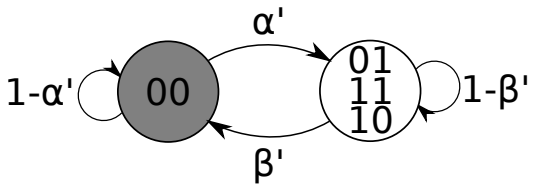


Figure 5.4. Equivalent Markov chain model for computation of time efficiency for partially randomized scheme where $\alpha' = 1 - (1 - \alpha)^2$ and $\beta = \frac{\alpha'\beta^2}{(\alpha+\beta)^2-\beta^2}$.

Since state $q = 00$ has not been merged with any other state, the transition probability on its self-loop is equal to the transition probability listed in (4.7) for the 4-state Markov chain in Fig. 5.2

$$1 - \alpha' = (1 - \alpha)^2, \tag{5.10}$$

and thus

$$\alpha' = 1 - (1 - \alpha)^2. \tag{5.11}$$

Furthermore, the probability for state $q = 00$ in the stationary distribution of the 2-state Markov chain has to be equal to the probability listed in (4.8) for the 4-state Markov chain in Fig. 5.2

$$\begin{aligned} \mu'_{00} &= \mu_{00} \\ \frac{\beta'}{\alpha' + \beta'} &= \frac{\beta}{\alpha + \beta}, \end{aligned} \tag{5.12}$$

and thus

$$\beta' = \frac{\alpha'\beta}{(\alpha + \beta)^2 - \beta^2}. \tag{5.13}$$

Substituting (5.11) in (5.13), we obtain

$$\beta' = \frac{[1 - (1 - \alpha)^2]\beta}{(\alpha + \beta)^2 - \beta^2}, \tag{5.14}$$

and then substituting (5.14) in (5.9), we obtain

$$E[\eta(K)] = 1 + \frac{T_{UP} + T_{DN}}{T} \cdot \frac{\frac{[1-(1-\alpha)^2]\beta}{(\alpha+\beta)^2-\beta^2} \ln\left(\frac{[1-(1-\alpha)^2]\beta}{(\alpha+\beta)^2-\beta^2}\right)}{1 - \left[\frac{[1-(1-\alpha)^2]\beta}{(\alpha+\beta)^2-\beta^2}\right]}. \tag{5.15}$$

□

Experimental results for the time efficiency obtained for Sun SPOTs are plotted alongside analytical results in Fig. 5.3(d). Similar to the case of the randomized scheme, we observe that the analytical results are slightly lower than the experimental ones. This behavior can be explained by the fact that we use a constant value for the set up and tear down costs T_{UP} and T_{DN}, whereas in the experiments these are random variables. Thus, the occurrences of values lower than the average have a stronger impact on the time efficiency than occurrences of values greater than the average.

An upper bound for the time efficiency can be computed using a formulation similar to the one employed for the connection delay of the randomized scheme in Section 4.3.3.

Theorem 14 (Upper bound on expected time efficiency)**.** *Given a partially randomized Markov chain-based duty cycling scheme, the upper bound on the expected time efficiency is*

$$E[\eta(K)] < \eta\left(\boldsymbol{h}'_{opt} \cdot \boldsymbol{p}\right), \tag{5.16}$$

where

$$\boldsymbol{p} = \left[0 \ \frac{\alpha\beta}{\alpha^2+2\alpha\beta} \ \frac{\alpha\beta}{\alpha^2+2\alpha\beta} \ \frac{\alpha^2}{\alpha^2+2\alpha\beta}\right] \tag{5.17}$$

is the re-weighted stationary invariant distribution of the pairwise Markov chain and $\boldsymbol{h}_{opt}$ *is the optimal solution to the linear optimization problem*

$$\begin{aligned} \min \quad & \boldsymbol{c} \quad \boldsymbol{h} \\ s.t. \quad & \boldsymbol{A} \quad \boldsymbol{h} = \boldsymbol{b} \\ & \qquad \boldsymbol{h} \geq \boldsymbol{0} \end{aligned} \tag{5.18}$$

with $\boldsymbol{c} = [1\ 1\ 1\ 1]$ *and*

$$\boldsymbol{A} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ (1-\alpha)\beta & -1+(1-\alpha)(1-\beta) & \alpha\beta & \alpha(1-\beta) \\ \beta(1-\alpha) & \beta\alpha & -1+(1-\beta)(1-\alpha) & (1-\beta)\alpha \\ \beta^2 & \beta(1-\beta) & (1-\beta)\beta & -1+(1-\beta)^2 \end{bmatrix} \tag{5.19}$$

$$\boldsymbol{b} = \begin{bmatrix} 0 \\ -1 \\ -1 \\ -1 \end{bmatrix} \tag{5.20}$$

(Sketch). Applying Jensen's Inequality to $E[\eta(K)]$, we obtain

$$E[\eta(K)] \leq \eta(E[K]), \tag{5.21}$$

where $E[K]$ is expected number of time slots until we hit state $q = 00$ given that the pair of nodes are in state $q \in \{01, 11, 10\}$. Substituting $\boldsymbol{h}'_{\text{opt}} \cdot \boldsymbol{p}$ for $E[K]$ in (5.21), we obtain

$$E[\eta(K)] < \eta\left(\boldsymbol{h}'_{\text{opt}} \cdot p\right). \tag{5.22}$$

□

In the next section, we present the results of the experiments we performed on Sun SPOT sensors for the proposed Markov chain-based duty cycling scheme and show how tuning the control parameters can bring about a solution with equal connection delay and improved energy efficiency.

5.3 Experimental Results for Partially Randomized Scheme

We performed a series of experiments with Sun SPOT sensors, and measured the connection delay, connection duration, time efficiency, and aggregate duty cycle,

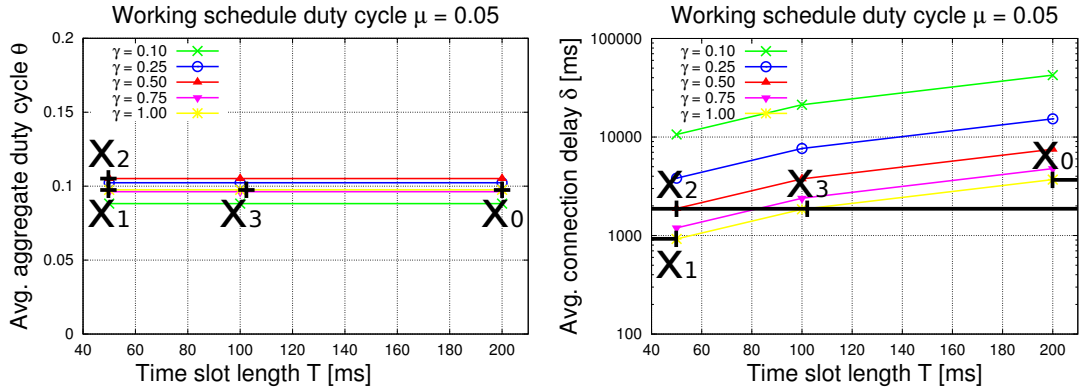
to validate the proposed Markov chain-based partially randomized scheme and the mathematical analysis. The considerations on the impact of hardware and software on the set up and tear down costs that we made in Section 4.4 when presenting the results of experiments for the Markov chain-based randomized scheme, apply to the experimental analysis of the corresponding partially randomized scheme as well.

We performed experiments of the partially randomized scheme for the same four different working schedule duty cycles ($\mu = 0.01, 0.05, 0.25, 0.50$), five memory coefficients ($\gamma = 0.10, 0.25, 0.50, 0.75, 1.00$), and three time slot lengths ($T = 50, 100, 200$ ms). According to the mathematical analysis in the previous section, in the experiments we consider pairs of wireless sensor nodes, and compute the one-hop connection delay and duration, and the time efficiency of the sensor nodes. Similarly to experiments for the randomized scheme, the working schedules consist of $6,000$ time slots. However, it is important to bear in mind that the resulting aggregate duty cycles are different, and thus the results in this section cannot be fairly compared to the ones for the randomized scheme. We plan to perform experiments with equal aggregate duty cycles in our future work. Nevertheless, this section offers a comparison between our partially randomized scheme and the mechanism in [59]. In fact, the latter is obtained as a special case of the earlier when the memory coefficient $\gamma = 1.00$ (*i.e.,* the states are *i.i.d.* random variables).

As far as the working schedule generation is concerned, the only difference in terms of complexity between the proposed partially randomized duty cycling scheme and the mechanism in [59] is the usage of two transition probabilities α and β and one bit of memory in the Markov chain, instead of the single probability μ for the Bernoulli trials yielding the independent identically distributed random variables.

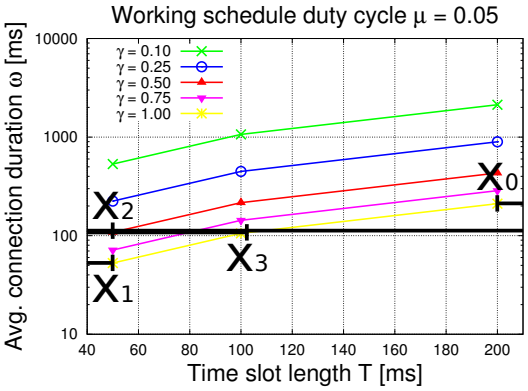
Fig. 5.5 displays results for aggregate duty cycle, connection delay, connection duration, and time efficiency vs. the time slot length for all memory coefficients and

duty cycle $\mu = 0.05$. In our discussion on the experiments, we focus on the results for points $\boldsymbol{x_0}$ through $\boldsymbol{x_3}$ listed in Table 5.1 and marked in Fig. 5.5 for wireless sensor nodes with duty cycle $\mu = 0.05$, although similar conclusions can be drawn for other duty cycles as well.

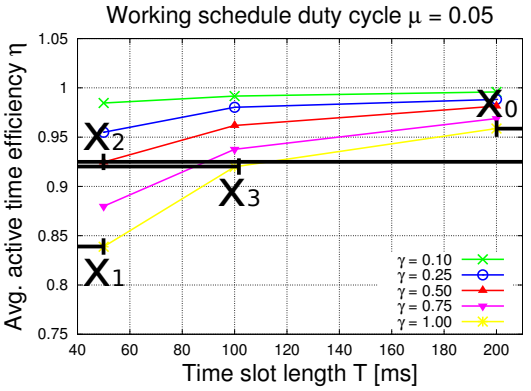


(a) Aggregate duty cycle, $\mu = 5\%$

(b) Connection delay, $\mu = 5\%$



(c) Connection duration, $\mu = 5\%$



(d) Time efficiency, $\mu = 5\%$

Figure 5.5. (a) Aggregate duty cycle, (b) connection delay, (c) connection duration and (d) time efficiency vs. time slot length for variable memory coefficient $\gamma = \alpha + \beta$ and duty cycle $\mu = 0.05$ for Sun SPOTs. By combining a reduction of the time slot length with a lower memory coefficient, the connection delay can be reduced, while not degrading the time efficiency. Similarly, the connection duration and time efficiency can be improved without affecting the connection delay, if the increase in the time slot length is accompanied by a higher memory coefficient.

Point	Memory coeff. γ	Time slot length T [ms]	Time eff. η (% diff.)	Conn. delay δ [ms] (% diff.)	Conn. dur. ω [ms] (% diff.)
$\boldsymbol{x_0}$	1.00	200	0.9587 ($\pm$0.00%)	3,695 ($\pm$0.00%)	211 ($\pm$0.00%)
$\boldsymbol{x_1}$	1.00	50	0.8388 ($-$12.51%)	924 ($-$74.99%)	53 ($-$74.88%)
$\boldsymbol{x_2}$	0.50	50	0.9248 ($-$3.54%)	1,880 ($-$49.12%)	108 ($-$48.81%)
$\boldsymbol{x_3}$	1.00	102	0.9207 ($-$3.96%)	1,880 ($-$49.12%)	107 ($-$49.29%)

Table 5.1. Time efficiency, connection delay, and connection duration for Sun SPOTs with duty cycle $\mu = 0.05$, aggregate duty cycle $\theta = 0.975$ and $T_{UP} = 11.50\ ms$ and $T_{DN} = 2.86$ ms.

Also for the Markov chain-based partially randomized scheme, we observe that all solutions $\boldsymbol{x_0}$ yield equal aggregate duty cycles, and thus we can compare the performance in terms of connection delay and time efficiency for a constant energy consumption.

Given the setting with time slot length $T = 200$ ms and memory coefficient $\gamma = 1.00$ ($\boldsymbol{x_0}$), the initial duty cycling scheme has an average connection delay $\delta = 3,965$ ms with time efficiency $\eta = 0.9587$. When the time slot length is reduced from $T = 200$ ms to $T = 50$ ms and the memory coefficient $\gamma = 1.00$ is upheld ($\boldsymbol{x_1}$), the duty cycling metrics are $\delta = 924$ ms and $\eta = 0.8388$. Besides featuring a lower connection delay, this setting also yields a lower time efficiency, which represents a negative side effect of the time slot length reduction.

Now, adopting our Markov chain-based duty cycling scheme with time slot length $T = 50$ ms and memory coefficient $\gamma = 0.50$ ($\boldsymbol{x_2}$) brings about a connection delay $\delta = 1,880$ ms, which represents a 49.12% reduction on the original value ($\boldsymbol{x_0}$), while keeping a time efficiency $\eta = 0.9248$. Although the memory-less setting with

$T = 102$ ms and $\gamma = 1.00$ ($\boldsymbol{x_3}$) based on [59] achieves the same connection delay $\delta = 1,880$ ms, it yields a time efficiency $\eta = 0.9207$, which is 0.44% below the 0.9248 efficiency of the Markov chain-based duty cycling scheme. It is easy to observe from Table III that the results for the connection duration are directly related to the connection delay, given that all settings have duty cycle $\mu = 0.05$.

Given a power consumption between 210-360 mW for a Sun SPOT [66], the Markov chain-based partially randomized scheme can reduce the energy consumption by 51-89 mJ per minute.

In the next section, we compare the performance of the proposed Markov chain-based randomized scheme and partially randomized scheme using the analytical model validated through these experiments.

5.4 Comparison of Randomized Schemes

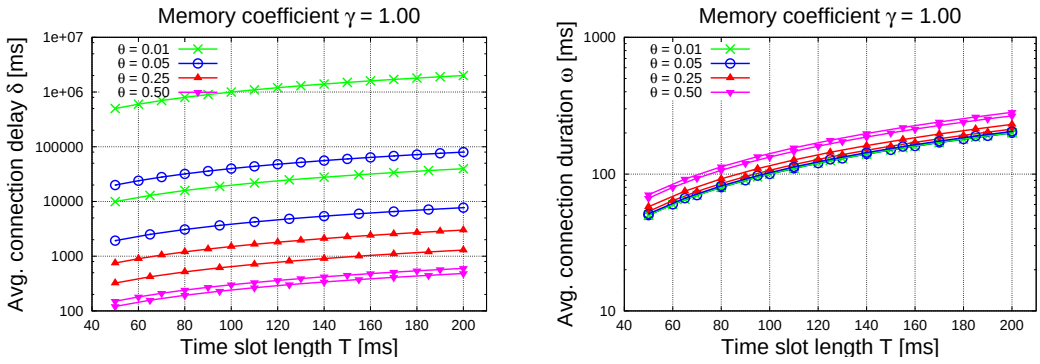
Given the analytical models for the randomized scheme and partially randomized scheme in Sections 4.3 and 5.2, it is worthwhile comparing the performances in terms of connection delay, connection duration, and time efficiency. In order to have a fair comparison, we set the schedule duty cycles μ's so that the two schemes yield equal aggregate duty cycles θ's, and hence feature equal energy consumption during operation. Furthermore, as it can be seen in Figures 4.1 and 5.1, both schemes make use of time synchronization (FSTP [64] in our experiments), thus keeping the comparison fair. The only extra cost for the partially randomized scheme is due to the working schedule distribution, which can be ignored for sufficiently long working schedules. In our experiments, each working schedule of $6,000$ time slots of 50 ms each lasts 5 minutes and can be compressed into 750 bytes. Even assuming a somewhat conservative data rate of 100 kbit/s for IEEE 802.15.4, one working schedule can be transferred in 60 ms. Assuming a maximum number of neighbors of 15, all working

schedules can be shared in less than 1 sec. This results in a 0.34% overhead for the partially randomized scheme as compared with the randomized one. The overhead is further reduced for longer working schedules or longer time slots. As discussed in the following, the improvement of the partially randomized scheme over the randomized one is in the order of 50%, thus making the overhead due to sharing of working schedules negligible.

The analytical results for the two schemes are plotted in Fig. 5.6. First, we observe that the time efficiencies in Fig. 5.6(c) are equal for a fixed time slot length, memory coefficient, and aggregate duty cycle. However, the connection delay and connection duration are not. The relative improvement (*i.e.,* decrease) in terms of connection delay is much higher for lower duty cycles (cfr. the distance between the green plots and the magenta ones in Fig. 5.6(a)). This is due to the fact that active instances for low duty cycle wireless nodes are unlikely to be overlapping. The same consideration motivates the symmetric behavior observed for the connection duration, where the relative improvement (*i.e.,* increase) is much higher for higher duty cycles (cfr. the distance between the green plots and the magenta ones in Fig. 5.6(b)). In this case overlapping active instances between neighboring nodes enable longer connections.

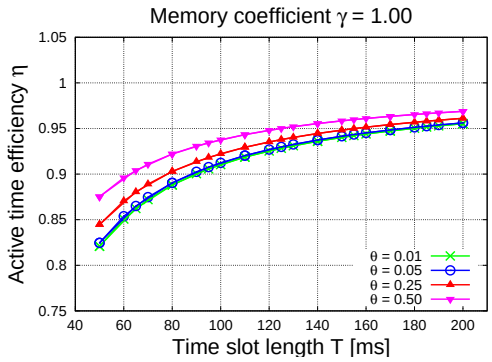
This analytical comparison demonstrates that partially randomized schemes are to be preferred over randomized schemes (whenever the working schedule distribution is not too expensive), since the connection delay and duration can be improved while affecting neither the aggregate duty cycle nor the time efficiency. Similarly, the aggregate duty cycle and the time efficiency could be improved, while keeping the connection delay and duration constant.

In the next section, we discuss the application of the analytical models to the design of optimal duty cycling schemes.



(a) Connection delay, $\gamma = 1.00$

(b) Connection duration, $\gamma = 1.00$



(c) Time efficiency, $\gamma = 1.00$

Figure 5.6. Analytical results for (a) connection delay, (b) connection duration and (c) time efficiency vs. time slot length for randomized scheme (10-ms spaced) and partially randomized scheme (15-ms spaced) with equal aggregate duty cycle $\theta \in \{0.01, 0.05, 0.25, 0.50\}$.

5.5 Optimal Duty Cycling for Wireless Sensor Networks

The expressions for the aggregate duty cycle θ, the connection delay δ, the connection duration ω, and the time efficiency η obtained in (4.10), (4.13), (4.19), and (4.21) for the randomized scheme and (5.1), (5.4), (5.6), (5.8) for the partially randomized scheme, respectively, can be employed to compute a solution for the working schedule duty cycle μ, the memory coefficient γ, and the time slot T that optimizes a function $c([\theta\ \delta\ \omega\ \eta]')$ of the performance metrics. The values of the control parameters that yield the optimal solution are provided to the wireless sensor nodes

that employ them in the generation of the working schedules, thus achieving a known level of performance in terms of aggregate duty cycle, connection delay and duration, and time and energy efficiency.

If the constraints in the optimization problem based on the mathematical model change (*e.g.,* the upper bound on the duty cycle is raised because the energy harvesting system is recharging the battery at a higher rate, or the set up and tear down costs are shorter), then the optimal solution for the control parameters (*i.e,* duty cycle, memory coefficient, and time slot length) should be recomputed as a solution with better performance in terms of connection delay, connection duration, and/or time and energy efficiency could be achieved. This operation can be performed on the wireless nodes themselves, or not. Since the optimization problem is rather complex, it might be best to solve it at the base station, and then send the new values for the parameters to the wireless nodes, that they can use to generate future working schedules. This solution is feasible because the constraints due to required lifetime and set up and tear down costs are likely to change quite infrequently (*i.e.,* in the order of several hours, if not days).

The performance improvement in terms of time and energy efficiency grows as the set up and tear down operations become more and more expensive (in terms of time and energy), which is the case in real-life WSN applications where wireless sensor nodes have to perform multiple operations when switching from deep sleep to normal operation mode, including communication stack update and sensor warm up.

We argue that this work can represent a baseline scheme against which successive works on duty cycling in WSNs that consider the time and energy efficiency metric could be evaluated.

5.6 Summary

In this chapter, we applied the proposed Markov chain-based solution to a partially randomized scheme and studied the performance via mathematical analysis and experimental validation. We showed that the energy efficiency of the proposed Markov chain-based partially randomized scheme can be improved while not affecting the connection delay, or vice versa.

The proposed schemes are much more effective in designing sustainable and green real-life WSNs applications, where the set up and tear down costs are much higher compared to the baseline in our experiments due to communication stack update and other operations.

We are currently conducting experiments with a set of realistic set up and tear down operations on a network of nodes, consisting of both Sun SPOTs and MEMSIC TelosBs [82]. Future work includes studying the end-to-end connection delay and analyzing the performance in terms of aggregate duty cycle for other communication modes besides the convergecast one considered in this chapter.

CHAPTER 6

FuseViz:

A FRAMEWORK FOR WEB-BASED DATA FUSION AND VISUALIZATION IN SMART ENVIRONMENTS

In recent years, pervasive computing and communications technology has enabled the development of applications in a wide spectrum of domains: from traffic and transportation [8] to smart environments [83], from health and well being [84] to fitness and sports [85], and from games and entertainment [86] to social networking [87], just to name a few. As depicted in Fig. 6.1, these domains have two important common features: *i)* diverse data sources, and *ii)* lay (application) users.

Figure 6.1. Smart environment scenario. In pervasive computing scenarios, valuable information for users is scattered in various kinds of data originating from diverse independent sources. A pervasive computing application should aggregate the relevant data from all sources in a meaningful way, and present them to lay users in a format such that it is easy for them to explore the data and obtain the necessary information. Redesign of Fig. 1 in [88].

Thanks to advances in pervasive computing, wireless mobile communications, and micro-electronic mechanical systems (MEMS) technology, data can be collected from a large array of sources spanning from wireless sensor nodes (*e.g.,* [62, 82]) and smartphones (*e.g.,* [89, 90]) to online databases [91]. To present a complete view of a domain (*e.g.,* smart home), an application often needs to rely on various data sources (*e.g.,* appliances and user location). Furthermore, the same data source (*e.g.,* user location) may contribute to applications in different domains (*e.g.,* smart home and road navigation). Therefore, it is important to develop a systematic solution for the *aggregation* of multiple data sources in pervasive computing applications. Features of data streams and sources [92], such as size, format, liveness, heterogeneity, dynamicity, and autonomy, pose several challenges that the aggregation mechanism should address.

Due to their pervasive nature, applications in the above-mentioned domains often feature an audience of lay users with no data analysis background. Therefore, it is important that the data be presented in a fashion that makes it easy for these users to analyze them intuitively, and extract the relevant information. *Visualization* is an effective solution for intuitive data analysis as it leverages the human vision capability to spot patterns and trends.

To better present our arguments about data and users, let us introduce an example which we use as the case study throughout this chapter, namely the Energy-Efficient Home (E^2Home) application. In a smart home scenario, lay users are interested in the electric energy consumption at their home, and would like to understand the patterns and trends in order to adjust their behavior, thus making the home more energy-efficient.

First of all, data from the electric energy meter are needed. However, these data alone are likely to be insufficient to understand how to change one's behavior to reduce

energy consumption. In fact, energy consumption data could be joined with the user location to understand where the user is, when high energy consumption occurs. The user could then try to reduce high energy consumption occurring when he/she is not home. For instance, the heating, ventilation, and air conditioning (HVAC) system could be better programmed to match the user's daily routine.

Notwithstanding the final set of data sources, we observe that they are likely to be diverse and autonomous from each other, while all being relevant to the problem, and thus instrumental to improve the energy efficiency of a smart home. In our case study, electric energy consumption data is programmatically retrieved via the Smart Meter Texas (SMT) Web site [93], and the user's location is collected using an Android [90] app on a Google Nexus One [94] smartphone.

One of the problems in the design of an application using data from multiple autonomous sources is that it is impossible to aggregate them by performing a join operation as defined on tables in a relational database. For instance, in the E^2Home application, the data streams from the SMT Web site and the Android smartphone app are unsynchronized. Not only it is impossible to synchronize the clocks of the two data sources, but also the sampling periods are different due to the nature of the quantities, and the instruments used to measure them. The energy consumption data are available for 15-min time slots from the SMT Web site, while the user's location is sampled with a 5-min period[1]. Therefore, it is impossible to join the data streams directly over the time dimension to obtain a *joint data stream* with energy consumption and user's location for each time.

[1]Clearly, both the 15-min time slot, and the 5-min period are the result of a trade-off between accuracy and cost, since sampling more frequently would incur in higher network traffic (SMT), or shorter battery life (Android app).

Even if the data streams relevant to the problem domain can be composed in a single joint data stream, static visualizations may not be sufficient to help a lay application user understand the data and extract the valuable information. In case of the E^{2}Home app, the user might be interested in the energy consumption when he/she is not home. However, a static time line with the energy consumption and a map with the user's location do not enable this kind of analysis, because data points in one chart (*i.e.,* the map) cannot be selected to see the corresponding value in the other dimension (*i.e.,* the time line). Therefore, interactive visualizations are needed, which support operations such as *panning, zooming,* and *brush-and-linking*.

While several applications [95, 96, 88, 92] provide static and also interactive visualizations, none of them addresses the problem of aggregating multiple autonomous data streams in a systematic fashion. Furthermore, even the few interactive visualizations are either customized for a specific application (*e.g.,* [95, 96]), or provide a limited interaction due to shortcomings in the re-rendering functions of the underlying visualization framework (*e.g.,* [88, 92]). This motivates our work.

In this chapter, we present FuseViz, a framework for the development of Web-based data aggregation and visualization in pervasive computing environments[2]. The objective of FuseViz is to exploit the full potential of available data sources in the application domain to the user's advantage by easing the ascent of the data-information-knowledge-wisdom (DIKW) pyramid [99] as depicted in Fig. 6.2. To achieve this goal, FuseViz relies on two basic features: *i) aggregation*, and *ii) visualization*.

To solve the problem of composing multiple autonomous data sources, we define the concept of aggregation. Aggregation is a generalization of the join operation

[2]The research work presented in this chapter has been published in the Proc. of the 9$^{\text{th}}$ IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS) [97] and the Proc. of the 3$^{\text{rd}}$ IEEE International Conference on Smart Grid Communications (SmartGridComm) [98].

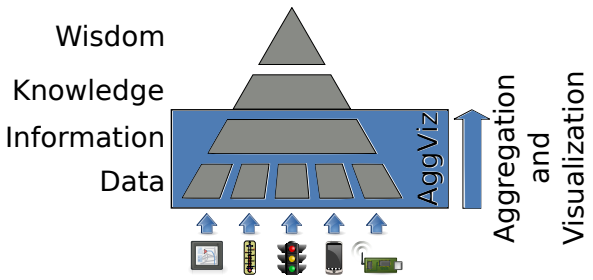


Figure 6.2. DIKW pyramid and the FuseViz framework. In the DIKW pyramid model, data are transformed into information, followed by information into knowledge, and finally knowledge into wisdom. The FuseViz framework uses aggregation and visualization to transform data from multiple sources in a pervasive computing environment into information for lay users.

according to which two or more data streams are joined along one or more dimensions (*e.g.,* time, position) to form a joint data stream. The aggregation function is not uniquely defined, but it depends on the domain and the application. For instance, in the E^{2}Home application scenario, energy consumption and user's location can be aggregated over time, thus computing an average value for each one of them or keeping all data points for each time interval.

In FuseViz-based applications, data from multiple sources are stored in CouchDB [44], a schemaless database, and then aggregated into a joint data stream using MapReduce [100] functions. To the best of our knowledge, the proposed aggregation mechanism is the first solution to join multiple autonomous data streams in pervasive computing environments.

Joint data streams are visualized on scalable vector graphics (SVG) [101] charts and maps in a Web page using D3 [102, 103], a JavaScript library to manipulate data-driven documents. The resulting SVG-based interactive visualizations are very responsive owing to element-specific re-rendering enabled by D3.

Due to aggregation and visualization in FuseViz, data from multiple autonomous sources can be transformed into valuable information for lay users in any application

scenario. To assess the proposed FuseViz framework, we developed several visualization applications, including the E^{2}Home app for the case study in this chapter. The interactive visualizations of the joint data stream featuring energy consumption and user's location enable the user to narrow down on high energy consumption time intervals when he/she is not home using brush-and-linking and panning and zooming. The user can then exploit this information to adjust his/her behavior, or tune home appliances in order to reduce the overall energy consumption.

Our analysis of the experimental results obtained over a 38-day period shows that a large share of the energy consumption (11.34%) is concentrated in a small fraction of the overall time (1.09%) while the user is not home. This implies that targeted actions in these situations (*i.e.,* user not home and high energy consumption) can bring about a significant improvement in the home energy efficiency. In fact, the energy efficiency of the home can be improved by more than 10%, if the user takes advantage of the precise information provided by the E^{2}Home application to better control the electric appliances.

The rest of the chapter is organized as follows. Section 6.1 summarizes the related work. Section 6.2 formalizes the requirements of the FuseViz framework, and discusses the related challenges. Section 6.3 offers a detailed description of FuseViz, and Section 6.4 presents the case study of the E^{2}Home application. Section 6.5 discusses the performance of FuseViz based on the E^{2}Home application, and compares it with that of other data storage and/or visualization frameworks. Finally, conclusions are drawn in Section 6.6.

6.1 Related Work

Over the last few years, there has been a growing interest on visualization in the context of pervasive computing and communications. We observe a shift from cus-

tomized solutions to more generic ones, and from static visualizations to interactive ones. Nevertheless, none of the existing systems for visualization of data from pervasive environments provides a solution for the aggregation of multiple autonomous data sources.

Biketastic [104, 96] and the Copenhagen Wheel [95] are domain-specific systems that offer several advanced visualizations, including map overlays of data (*e.g.,* pollution levels, traffic congestion, and road conditions) collected by users. Although users can interact with visualizations to a certain extent, both projects appear to be customized solutions with no support for quick addition of data sources or visualizations, thus making it impossible to use them as data visualization frameworks in any other application domain.

Sensor.Network [88, 105] is a data storage and exchange platform for the Internet of Things (IoT) that offers a ReSTful (representational state transfer) [33] API for users, or devices/services acting on their behalf, to manage data streams as well as insert or retrieve data. As such, Sensor.Network is successful in creating a common place for storing data streams in different domains. It also provides several visualizations, including maps and scatter charts, but these are not interactive and data streams within Sensor.Network cannot be aggregated into a joint data stream to be visualized.

Earlier we developed SNViz [92] as a visualization framework for data streams in Sensor.Network [88]. SNViz caters to lay users by making it easy to visualize different dimensions of a data stream on charts and maps and providing support for panning, zooming, and brush-and-linking. However, interactivity suffers from the intrinsic shortcomings of using Protovis [106], as re-rendering of subsets of elements in a visualization is not a top-level operation in Protovis, but is achieved using local variables that severely slow the process, and may even cause inconsistencies in the

underlying data structures [102]. Furthermore, SNViz does not address issues arising from autonomous data sources as it is tightly coupled with Sensor.Network, nor data streams in Sensor.Network can be aggregated into a joint data stream.

On the other hand, FuseViz addresses the issues related to existing data visualization solutions in pervasive computing and communications, including diverse data streams and static visualizations. The outcome is a novel framework for Web-based aggregation and visualization of data from multiple autonomous sources.

In the next section, we formalize the requirements that guide the development of FuseViz, and discuss the challenges.

6.2 Requirements and Challenges

The FuseViz framework for analysis-oriented visualization by lay users in pervasive computing environments has the following requirements: *i)* support for Web access; *ii)* modular and extensible data sources and visualizations; and *iii)* embeddable, interactive, and responsive visualizations.

First of all, FuseViz-based applications should be accessible from any device, including smartphones and tablets, because in a pervasive computing scenario users likely want to access the relevant information independent of the devices currently at their hands.

Second, FuseViz should be modular and extensible both in terms of data sources and visualizations. An FuseViz-based application should be easily assembled using only the necessary data sources and visualizations (modularity), and data sources and visualizations should be easily added whenever an application domain requires them (extensibility).

Finally, FuseViz-based applications should be interactive, responsive, and embeddable in existing Web pages. Responsive visualizations make it more appealing

for users to interact with them to explore the patterns and trends in the data. Furthermore, enabling features like embedding of visualizations in Web pages makes it easy to share them with other interested users via social networking applications, or create mash ups on blogs.

In order to develop an effective solution for analysis-oriented visualization, several challenges posed by the nature of data and users need to be addressed. In this study, we argue that major challenges are due to the following features of the data streams and sources in a pervasive computing environment: *i)* size, *ii)* liveness, *iii)* heterogeneity and dynamicity, and *iv)* autonomy.

Large data streams consisting of several (hundreds of) thousands of data points, each carrying values along multiple dimensions, need be efficiently stored, aggregated, transferred, and rendered in the interactive and responsive visualizations in a Web browser. For instance, even in a simple application such as E^2Home, 17,856 samples/month per (user, home) pair are being collected.

Not only are data streams large, but they are also likely to be live as new data points are added while users take advantage of the visualization application. New data points should be stored and aggregated, and visualizations should be updated to incorporate the new data without disruption to the user currently interacting with them. In the E^2Home app, one new sample is generated every 3 min 45 sec on average for a (user, home) pair.

Since data streams originate from multiple sources, they are heterogeneous in terms of such features as data types, sampling frequencies, and accuracy. Data stream features are also likely to change over time because they are not usually under the control of those developing the visualization applications. In the E^2Home app, developers do not control the design of the SMT Web site and the formats used for

energy consumption data. Therefore, storage, aggregation, and visualization should be designed to adapt to heterogeneous and dynamic data streams.

Relevant data for a given application originate from autonomous data sources, whose primary goal might not be to support a specific application. For instance, the organization managing the SMT Web site is unaware of the use of electric energy consumption data made by the E^{2}Home app. Therefore, FuseViz should enable aggregation of these autonomous data sources also in case of temporary disruptions to the data streams.

Finally, although lay users do not have a background in data analysis, they can easily spot patterns and trends when data are visualized. However, even if the patterns and trends can be spotted, having a large number of data points (*e.g.,* $>$ 17k samples/month in the E^{2}Home app) on a visualization can severely impair this process unless the user can explore the data highlighting relationships and focusing on subsets. FuseViz-based visualizations should help this process by providing intuitive interaction methods to explore the (joint) data streams.

In the next section, we describe the architecture of the FuseViz framework, and highlight how it fulfills the requirements and addresses the challenges that we outlined.

6.3 Proposed FuseViz Framework

Before describing the FuseViz framework, let us introduce two important building blocks: CouchDB [44] and D3 [102]. CouchDB is a distributed schemaless database with a ReSTful JSON (JavaScript Object Notation) API to manage databases and documents over HTTP. CouchDB documents are JSON-formatted and can support MIME (Multipurpose Internet Mail Extensions) attachments for non-text data types. Server-side JavaScript *validation functions* can be designed to check documents

being pushed to a database. Once stored in a CouchDB database, documents can be automatically copied to other databases using CouchDB's replication mechanism. Documents in a CouchDB database can be queried and indexed using server-side JavaScript MapReduce functions called *views*. The output of MapReduce functions can be further processed using *list functions* before being sent as part of the HTTP response. A CouchDB list is a server-side JavaScript function that takes the JSON array resulting from the MapReduce function as input, and transforms it into other formats (*e.g.,* HTML or RSS XML), or performs reduce-like operations on the data points.

D3 is a JavaScript library for manipulating documents based on data. Data in the form of JavaScript arrays, thus including JSON documents, can be associated to elements in the Document Object Model (DOM), such as SVG and HTML ones. The properties of these DOM elements can then be set based on the value of the corresponding object in the data array. D3 also supports dynamic data sets via so-called transformations, as DOM elements can be created, updated, or deleted when the elements in the data array are added, edited, or removed.

As discussed in the rest of the chapter, CouchDB and D3 are instrumental in providing aggregation and visualization in FuseViz as they help us address the challenges due to data and users in pervasive computing applications.

6.3.1 Architecture

Figure 6.3 portrays the architecture of the FuseViz framework that consists of the following components: *collection, storage, aggregation,* and *visualization.* In the following, these are described in details.

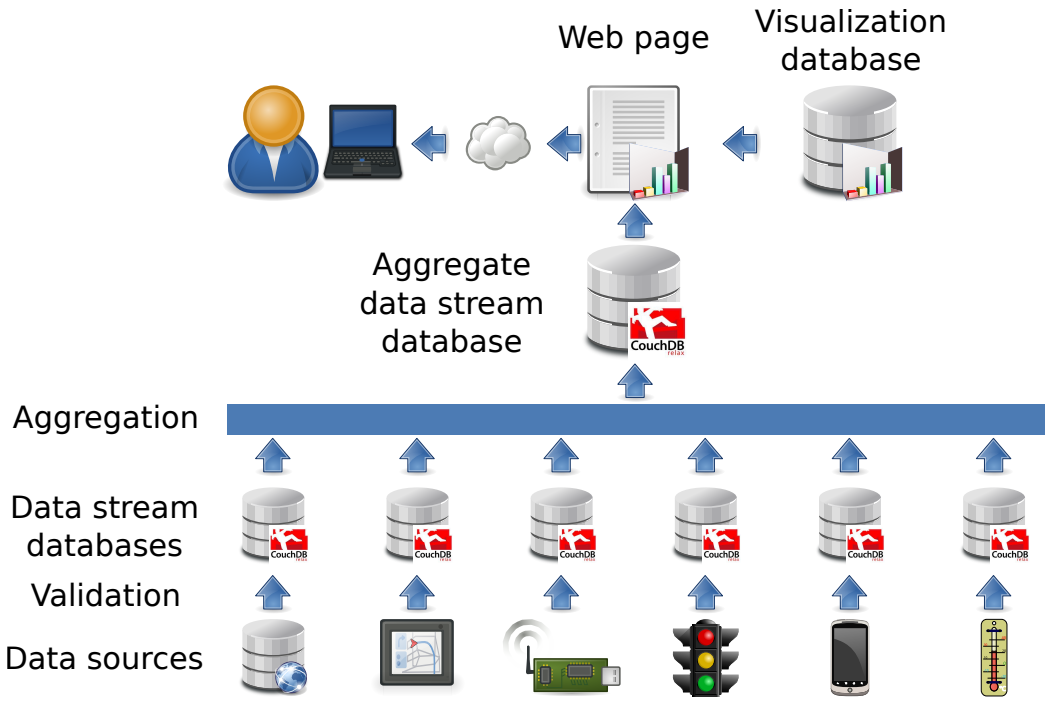


Figure 6.3. Architecture of FuseViz-based application. Data from heterogeneous sources are *i)* validated and pushed into CouchDB databases; *ii)* aggregated into a single joint data stream using MapReduce functions; and *iii)* presented to the user in a Web page using interactive visualizations.

6.3.1.1 Data Collection

Data from all relevant sources for the visualization application are imported into a CouchDB database. If the source already stores the data in such a database, then they can be easily imported using CouchDB's replication mechanism. Otherwise, the data collection application should directly post the data points to CouchDB, or a bridging application should be implemented.

Since CouchDB presents a ReSTful JSON API to manage documents, data points should be formatted using JSON to be pushed to CouchDB using HTTP. As we show in the E^{2}Home application, these constraints do not pose major hurdles as most programming languages feature HTTP and JSON libraries.

As previously mentioned, validation functions can be implemented to check the data points being pushed into the database. This is particularly important given that visualization applications aggregate data from third-party sources (*e.g.,* SMT Web site in our E^{2}Home application), which cannot be trusted as to the format of their data.

An important design decision is whether data points from different sources should be imported into a single database, or not. We argue that each data stream should be imported into its own database because it could then be used to feed multiple visualization applications. For instance, the user's location is likely to be used in other applications beyond the E^{2}Home app. This solution has also the positive side-effect of simplifying validation function design, as specific functions for each data source can be implemented. A single validation function for energy consumption and user's location data in the E^{2}Home application would not support modularity and extensibility.

Challenges due to data and sources are easily addressed using CouchDB. In fact, CouchDB's Erlang-based HTTP engine easily scales up to handle heavy loads of requests [107], thus enabling data collection for multiple autonomous, large, and live data streams. Furthermore, CouchDB's ReSTful JSON API provides a straightforward solution to import data from heterogeneous and autonomous sources.

6.3.1.2 Data Storage

Once data streams are pushed to a CouchDB database, an important design decision is whether multiple data points should be stored in a single CouchDB document, or not. We argue that the best engineering solution is to have one data point per document, because this makes MapReduce function design straightforward. Furthermore, the data collection would be more cumbersome, if multiple data points per

document were allowed. In fact, validation functions should be designed to allow for variable number of data points in a HTTP request, and stored documents should be first retrieved by the data collection application or bridge, updated with the new data, and then pushed into the CouchDB database again.

Using CouchDB, large, heterogeneous, and dynamic data streams from autonomous sources can be easily handled by FuseViz. As long as a data point passes validation, it can be stored in the CouchDB database for its data stream, thus avoiding problems incurred into if using relational databases with a fixed schema.

6.3.1.3 Data Aggregation

Once data streams used in the FuseViz-based application are stored into CouchDB databases, they need to be aggregated into a joint data stream to be visualized on a Web page. Aggregation is needed because autonomous data sources are likely to generate data streams with different frequencies and offsets, or unnecessary dimensions. For instance, in the E^2Home data sources, data points for user's location feature a 5-min period, while electric energy consumption data points feature a 15-min period. In other application scenarios, data could be collected according to a threshold (*e.g.,* a new data point may be generated when the distance between the current location and the one in the last data point is above a preset value), thus with no regular period.

Data aggregation consists of three phases. First of all, relevant data streams are imported from their specific CouchDB databases into another database using CouchDB's replication mechanism. By leaving the CouchDB databases storing the component data streams unaffected, these can be used in multiple FuseViz-based applications.

Second, a MapReduce function in the CouchDB database is used to sort all data points, eventually removing unnecessary dimensions, from all data streams according to a key value. This is usually the field used for the aggregation of the autonomous data streams. In most applications, the key is likely to be some kind of timestamp associated with the data point. However, time is not the only dimension along which aggregation is performed. Another common field for aggregation is location.

Finally, the sorted data points are aggregated using a CouchDB list. Depending on how complex the aggregation is, a list function might be needed, or the reduce function in the MapReduce view might be sufficient to generate the joint data stream.

The result of these three steps is a joint data stream, which can be retrieved via a HTTP GET to the list uniform resource identifier (URI), or the view URI if the list function is not used.

The data aggregation component addresses several challenges due to the features of data streams. First of all, the MapReduce paradigm is well-suited to process heterogeneous and dynamic data streams. Not only can specific operations be designed for different data streams being aggregated, but also they can be easily updated.

Second, the result of MapReduce functions is stored in the database, so that they do not have to be recomputed for each request. Therefore, the computational load on CouchDB and the latency of HTTP responses are minimized even in case of large data streams.

Finally, the result of a MapReduce function is recomputed on-the-fly for a HTTP request, if there exist new data points in the database due to live data streams.

6.3.1.4 Data Visualization

Once data from multiple sources have been aggregated into a joint data stream, they are ready to be visualized in a Web page. Since the objective of a Web-based data visualization application is to be accessible by the users anytime anywhere, it is important that the technologies used in its implementation are supported not only on laptops and desktops, but also smartphones and tablets. However, these handheld devices often miss components that are given for granted on laptops and desktops (*e.g.,* Adobe Flash on Apple WebKit). We decided to use HTML, JavaScript, SVG and cascading style sheets (CSS) as these technologies are standards and are already available on a wide array of devices, or are likely to become available in the near future. (SVG is available on Firefox Mobile Web browser on Android and the Apple WebKit on iPhone, but not in the Android default Web browser.)

The visualization subsystem depicted in Fig. 6.4 relies on D3 to manipulate visualizations based on the data in the joint data stream. So far we implemented the following visualizations: scatter chart, line chart, focus and context time line, and map. In all visualizations, data points are depicted using SVG circles. The fill color is red or blue depending on whether the data point has been selected or not. The focus and context time line (*cfr.* Fig. 6.7) is a composite visualization where the time interval selected in the bottom time line chart (context) is portrayed in the top time line chart (focus). Furthermore, the user can slide the selection to observe evolution over time. The map visualization (*cfr.* Fig. 6.7) is implemented using an SVG overlay on top of Google Maps.

Use of D3 implies large, live, and heterogeneous data streams can be rendered on interactive visualizations for lay users. First of all, D3 allows for editing of subsets of DOM (*e.g.,* SVG) elements, so that only those elements affected by user-generated

events (*e.g.,* brush-and-linking) are updated. Similarly, elements can be added or removed to visualizations of live data streams without having to re-render all other elements. Furthermore, D3's agnostic approach to standards does not restrict visualizations to one technology (*e.g.,* SVG). As a result, visualizations including not only scalar data, but also multimedia ones (*e.g.,* images) can be incorporated, thus broadening the applicability of FuseViz to more domains in pervasive computing. Finally, D3's selection mechanism greatly simplifies the implementation of panning, zooming, and brush-and-linking: three features instrumental to transform raw data into valuable information for lay users.

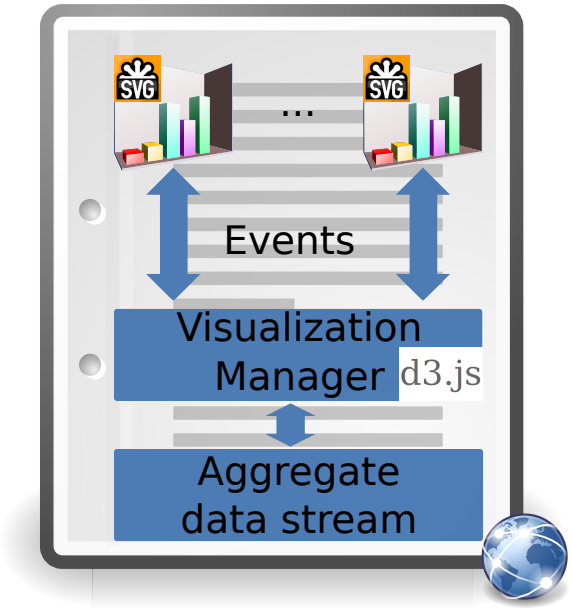


Figure 6.4. Architecture of data visualization subsystem. Aggregate data are visualized on SVG-based charts and maps using JavaScript and D3. A JavaScript-based visualization manager processes user-generated events such as panning and zooming and brush-and-linking, and updates the visualizations accordingly.

6.3.2 Application

A Web-based analysis-oriented data visualization application can be quickly developed on top of the visualization and aggregation components in FuseViz. As depicted in Fig. 6.4, a visualization application consists of plain HTML and JavaScript in a Web page.

The joint data stream is retrieved from the CouchDB database via a HTTP GET request to the list or view URI. The visualizations are loaded as scripts in the Web page along with the visualization manager. Visualizations are laid out on the Web page and registered with the visualization manager responsible for passing the data points to them, and relaying events, such as brush-and-linking, among them.

New data points in live data streams are retrieved using AJAX (asynchronous JavaScript and XML), and are seamlessly visualized using D3's support for dynamic data sets via transformations. Since CouchDB uses Etags [108], bandwidth consumption due to periodic polling of data stream lists or views by client Web pages is minimized.

Due to CouchDB's support of JSON padding (JSONP), visualization applications can also be embedded in other Web pages. In JSONP, data streams are loaded as scripts instead of using AJAX, thus working around the restrictions on cross-site scripting imposed by the same-origin policy (SOP) [109] in Web browsers. The only drawback of using JSONP to embed visualizations in other Web pages is that the Web page has to be reloaded to show new data points in live data streams.

6.3.3 Implementation

FuseViz is implemented as a set of CouchApp [110] applications. CouchApp is a toolkit on top of CouchDB that enables the development of Web-based applications. In a CouchApp, all data, logic (*i.e.,* validations, views, and lists), and presentation

(*i.e.,* HTML pages and related documents such as scripts and style sheets) are stored in a CouchDB. The data are stored in regular documents, while logic and presentation are stored in *design documents*. A CouchApp Web page is accessed via any Web browser. As far as CouchDB is concerned, it is just serving a document over HTTP. The Web page loads the related scripts and style sheets from CouchDB, or other sources (*e.g.,* Google Maps), and responds to user's actions by creating, reading, updating, and deleting data documents in the CouchDB database.

In FuseViz, we have implemented sample CouchApps for source and joint data streams, visualization, and application. The sample CouchApps feature also default validation, view, and list functions, when applicable. As we demonstrate in the next section, developers can easily implement and deploy a data visualization application using these CouchApps as building blocks.

6.4 Case Study: E^{2}Home

To showcase the strengths of FuseViz, we developed a few data visualization applications, including the E^{2}Home application for the case study described above. To recap, the objective of the E^{2}Home application is to provide lay users with valuable information about their home electric energy consumption, so that they can take actions to make the home more energy-efficient. We argue that electric energy consumption data alone are not sufficient to achieve this goal, and other data should be also incorporated, such as the user's location.

The architecture of the E^{2}Home application is depicted in Fig. 6.5. It follows the template set by FuseViz, and thus consists of data collection, storage, aggregation, and visualization on a Web page. The electric energy consumption data are retrieved from the SMT Web site [93], and are imported to the CouchDB data stream database using a Python script. The Python script periodically accesses the Web site with

the user's credentials, and downloads a CSV (comma-separated value) file with the electric energy consumption data. The downloaded CSV file is then processed into a JSON document conforming to the CouchDB data stream validation function, and uploaded to the CouchDB database via HTTP. Clearly, the data collection operation would be much simpler and secure, if SMT offered a ReSTful API, and supported *de facto* standard authentication mechanisms such as OAuth [111].

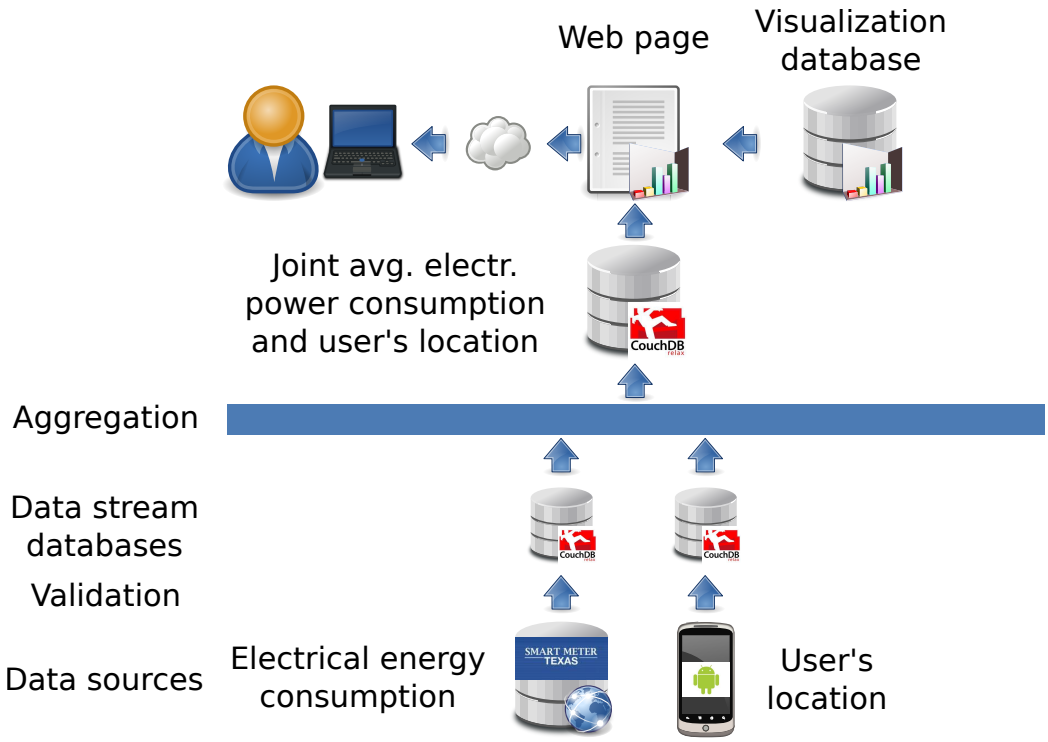


Figure 6.5. Architecture of E^{2}Home application. Electric energy consumption from the Smart Meter Texas Web site and user's location from an Android app on a Google Nexus One smartphone are *i)* stored in CouchDB databases; *ii)* aggregated into a joint data stream using 15-min time slots; and visualized using the focus and context time line (average power consumption vs. time) and map (user location). Other relevant data sources that may be aggregated are the locations of other tenants, and the weather (*e.g.,* temperature) in the area.

The user location is retrieved using an Android app developed for E^{2}Home, and then posted to the CouchDB data stream database. The user location is sampled using available data sources, such as GPS (preferred) and Wi-Fi, in Android. If Internet access is not available, location data are temporarily stored on the device, and then uploaded whenever a connection is available. In our experiments, the Android app runs on a Google Nexus One, but other Android devices could be employed.

Sample data points for energy consumption and user location are portrayed in Figures 6.6(a) and (b), respectively. Recall that CouchDB requirements feature `_id` and `_rev` fields. Additional common fields are `type` and `timestamp`. The first one is used to easily distinguish between energy and location data points in the MapReduce function, while the latter stores the time instant to which the data point refers. (In the case of the electric energy consumption data, this is the start time of the 15-min time slot during which the energy consumption stored in the `kWh` field occurred.) Since the two data streams are unsynchronized, non-trivial processing is needed to aggregate them into a joint data stream ready for visualization. User authentication is not the focus here.

The two data streams are replicated to a common CouchDB database for which a view and a list function have been defined. In the view, there exists only the Map part of the MapReduce function. This takes data points of either data stream as input and indexes them by timestamp. The list function reads the ordered data points one by one and groups them into k-min time slots. It then computes *i)* the average power consumption (measured in kW) over that time slot from the sum of all energy consumption (measured in kWh) data points over the same period; and *ii)* the location centroid from all the locations recorded for that time slot. Finally, it outputs the data point for the joint data stream as depicted in Fig. 6.6(c).

```
{
   "_id": "d95da3a86db2759bceddad095315112d",
   "_rev": "1-95b7959197d558453a29fbf5e9c0fe29",
   "type": "energy",
   "timestamp": "2011-08-31T16:30:00.000Z",
   "kWh": 0.029
}
```


(a) Energy consumption


```
{
   "_id": "ffaea3f27dc1fac90296edfaf6a54896",
   "_rev": "1-466e07bd7da9d6313b73a9aa968739ba",
   "type": "location",
   "timestamp": "2011-08-31T16:41:54.000Z",
   "latitude": 32.732648849487305,
   "longitude": -97.11329340934753
}
```


(b) User location


```
{
   "_id": "d95da3a86db2759bceddad095315112d",
   "type": "avgpowerlocation",
   "timestamp": "2011-08-31T16:30:00.000Z",
   "latitude": 32.732648849487305,
   "longitude": -97.11329340934753,
   "kW": 0.116
}
```


(c) Joint avg. power consumption and user location

Figure 6.6. JSON documents for (a), (b), and (c) joint power consumption and location in the E^{2}Home app.

The E^{2}Home application is accessed via a Web page currently featuring two visualizations: *i)* a focus and context time line, and *ii)* a map on which the data points retrieved from the remote CouchDB live joint data stream database are plotted. For each data point, the average power consumption is plotted on the time line, while the average location is plotted on the map. A screenshot of the visualizations is portrayed in Fig. 6.7.

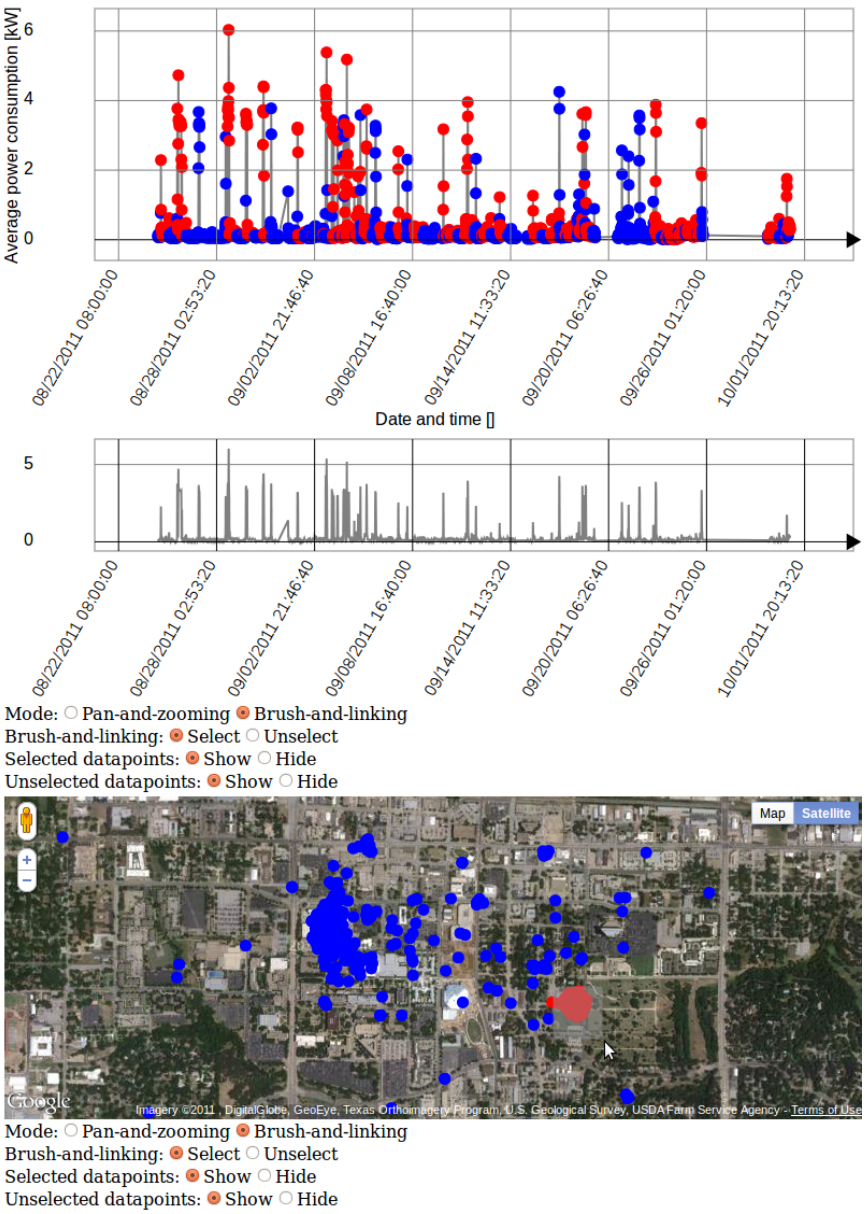


Figure 6.7. Brush-and-linking on E^{2}Home data. The user can select (brush) data points on the map corresponding to time slots when he/she was home (red data points inside gray rectangle in bottom chart), and then observe the average power consumption during the selected time slots (linking). Significant levels of average power consumption are observed also when the user is not home (blue data points > 2 kW in top chart). This could be due to wasteful consumption (*e.g.,* A/C on when nobody is home), or not (*e.g.,* washer on). Incorporating additional data sources such as all tenants' locations and energy consumption of individual appliances would provide even more valuable information.

6.5 Discussion

We argue that FuseViz fulfills the requirements, including providing valuable information for lay users from raw data in diverse application scenarios. The E^{2}Home case study is an example of this achievement. Figure 6.7 showcases how lay users can gain information from the interaction with visualizations of the aggregation of user location and electric energy consumption data. This information can then be used to decide what actions to take to improve the energy efficiency. As such, FuseViz provides a generic framework for the development of analysis-oriented visualizations that goes beyond customized solutions such as Biketastic [96] and Copenhagen Wheel [95].

We perform the following analysis of the experimental data to measure what impact the FuseViz-based E^{2}Home application can have on making a home more energy efficient. The electric energy consumption for the 38-day period between 08/24/2011 and 09/30/2011 is reported in Table 6.1. As it can be seen on Fig. 6.8, the 32 data points with high average power consumption (> 2 kW) have a disproportionate impact on the overall energy consumption. Although existing applications such as Google PowerMeter [112] and SMT [93] provide a time chart for the electric energy consumption, they do not include the contextual information (*i.e.,* user location) needed to help the user easily identify the exact times when high average power consumption occurred while he/she was not home.

Now, using the information provided by the E^{2}Home application, the user knows exactly at what times the high energy consumption occurred when he/she was not home. Therefore, he/she can take precise actions to remove this wasteful consumption in the following days. If this is the case, the 32 data points accounting for wasteful high energy consumption will be affected in the future. To compute the projected energy consumption for the following 38-day period, we substitute the 0.772 $\frac{\text{kWh}}{\text{data point}}$

Table 6.1. Electric energy consumption 08/24/2011 – 09/30/2011. The joint data stream covers 30 days 13 hours instead of 38 days because the user location was not available for short time periods when the smartphone was turned off. The (Not home, > 2 kW) and (Not home, ≤ 2 kW) sets consist of unselected data points (blue) in Fig. 6.7 with average power consumption above 2 kW, and below 2 kW, respectively; the (Home) set contains selected data points (red).

Data point set	No. of data points	Time [dd:hh:mm]	Total energy [kWh]	Avg. energy $\left[\frac{\text{kWh}}{\text{data point}}\right]$
Not home, > 2 kW	32	00:08:00	24.71	0.772
Not home, ≤ 2 kW	1,530	15:22:30	49.438	0.032
Home	1,370	14:06:30	143.817	0.105
All	2,932	30:13:00	217.965	0.074

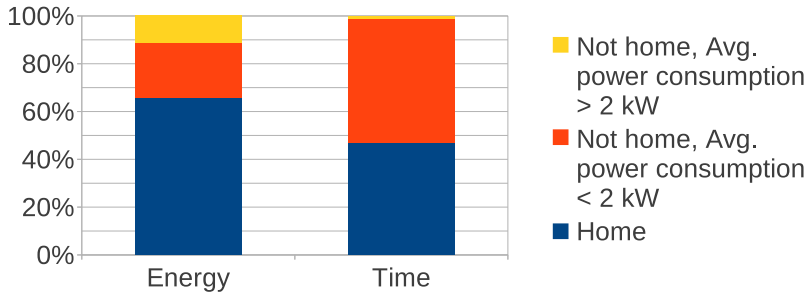


Figure 6.8. Observed energy consumption 08/24/2011 – 09/30/2011. While the 32 data points with average power consumption above 2 kW when the user is not home cover only 1.09% of the time, they account for 11.34% of the energy consumption. If this information is presented to the user via the visualizations in Fig. 6.7, he/she can take necessary actions to reduce the energy consumption associated with these relatively few data points, thus having a major impact on the overall energy efficiency of the home .

average energy consumption of the 32 high energy consumption data points with the 0.032 $\frac{\text{kWh}}{\text{data point}}$ average energy consumption of the 1,530 low energy consumption data points. As depicted in Fig. 6.9, the total energy consumption for these 32 data points decreases from 24.71 kWh to 1.034 kWh, thus bringing about a projected overall energy efficiency improvement of 10.86%.

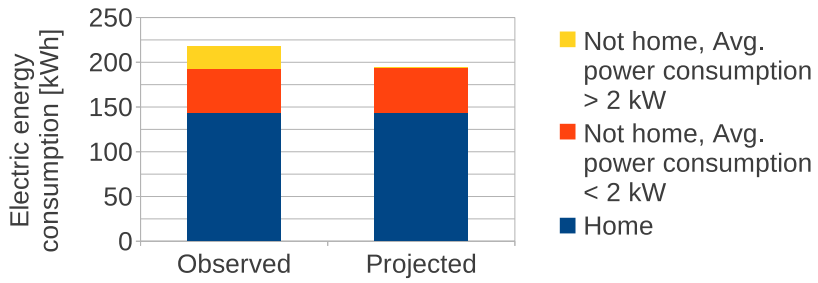


Figure 6.9. Projected energy consumption 08/24/2011 – 09/30/2011. If the user reduces the energy consumption associated with the 32 data points with average power consumption above 2 kW when he/she is not home, then the overall energy consumption for a 38-day period goes from the observed 217.965 kWh to the projected 194.289 kWh, which corresponds to an energy efficiency improvement of 10.86%.

The projected energy efficiency improvement can be directly computed in the E^{2}Home app, and be presented to the user. Clearly, additional data sources such as appliance-specific meters would further enhance the quality of the information in terms of where the consumption occurred, thus resulting in a more accurate estimation of the achievable energy efficiency improvement.

FuseViz visualizations are much more responsive since data rendering and re-rendering are much faster than in SNViz. This performance improvement is due to the use of D3 instead of Protovis [106], as D3 directly supports editing of single DOM elements, while Protovis does not, and even workarounds such as the ones employed in SNViz yield longer delays.

FuseViz supports open development of data visualization applications, since any set of data sources can be used and aggregated into a joint data stream. This is impossible in custom solutions such as Copenhagen Wheel or Biketastic built around one application scenario, and also in Sensor.Network or SNViz as those systems do not support data stream aggregation. To the best of our knowledge, this is the first solution for aggregation of autonomous data streams proposed for pervasive computing environments.

Finally, other features of FuseViz, such as JSONP-support and use of well-established standards, such as SVG and HTTP, make it very appealing for the embedding of FuseViz-based data visualization applications on social networking Web sites and blogs. We argue that this feature is likely to further enhance the value of the data as these can be viewed and manipulated in the context of other related data. For instance, this could be the case of a Web page aggregating the electric energy consumption data of a community of friends on a social networking Web site.

6.6 Summary

In this chapter, we presented a novel framework, called FuseViz, for the development of Web-based data aggregation and visualization applications in pervasive computing environments. We analyzed the requirements and challenges of such a system, and provided a thorough description of the proposed solution. To demonstrate the capability of FuseViz, we developed E^2Home, a data visualization application for smart home energy consumption, and showed how lay users can change their behavior to improve the home energy efficiency by more than 10% based on the valuable information presented by the FuseViz-based application.

We are currently working on the incorporation of multimedia data streams, such as images and audio/video, in the FuseViz framework. Future work also includes

an experimental study with a group of users to quantitatively assess the impact of FuseViz-based applications into improving user's behavior by providing them with valuable information from multiple data sources.

CHAPTER 7

CONCLUSIONS

In this dissertation, we report on our research work performed in the areas of wireless sensor networks and smart environments and the energy-efficient protocols and systems we developed. After an introduction into wireless sensor networks and the very important communication stack, we presented our proposed solutions. First, we introduced a localization protocol for dense wireless sensor and actor networks, and analyzed it using methods based on the coupon collector's problem and the theory coverage processes, as well as simulations.

Second, we investigated the energy efficiency of existing randomized protocols for duty cycling in wireless sensor networks and pointed out their trade-off between energy efficiency and connection delay using mathematical analysis and experiments on Sun SPOT sensors. Following our observation, we proposed a Markov chain-based randomized algorithm that improves energy efficiency, while not affecting connection delay, or vice versa.

Finally, we shifted our focus from wireless sensor networks to smart environments, and proposed a framework for data fusion and visualization called FuseViz. We used the FuseViz framework in the development of E^2Home, a Web-based application for improving energy efficiency in homes. Our experimental results in houses and apartments show that E^2Home can help residents lower their electricity consumption by 10% while keeping their current standard of living.

To conclude, we argue that after more than a decade of research wireless sensor networks and smart environments are slowly making their appearance in the real

world. Thanks to the penetration of smartphones, the deployment of smart meters, and the growing interest of homeowners and residents, more applications leveraging these technologies are likely to be developed in the upcoming years. However, several problems, including the definition of a system-level energy model for homes, are still open, and more research is needed to bring about effective solutions.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[3] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware Design Experiences in ZebraNet," in *Proc. of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004, pp. 227–238.

[4] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proc. of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004, pp. 13–24.

[5] Dust Networks, "Dust Networks Applications," 2010. [Online]. Available: http://www.dustnetworks.com/applications

[6] G. Tolle, D. Gay, W. Hong, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, and P. Buonadonna, "A macroscope in the redwoods," in *Proc. of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005, p. 51.

[7] A. R. Silva and M. C. Vuran, "Development of a Testbed for Wireless Underground Sensor Networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, pp. 1–14, 2010.

[8] INRIX Inc., "INRIX Traffic," 2011. [Online]. Available: http://www.inrixtraffic.com

[9] G. Ghidini, S. K. Das, and D. Pesch, "Sensor network communication protocols for greener smart environments," in *Design Technologies for Green and Sustainable Computing Systems*. Springer, 2013.

[10] IEEE 802.15 Task Group 4 (TG4), "IEEE Standard 802.15.4-2011," 2011.

[11] G. Montenegro, N. Kushalnagar, J. W. Hui, and D. E. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," 2007. [Online]. Available: http://datatracker.ietf.org/doc/rfc4944

[12] J. W. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," 2011. [Online]. Available: http://datatracker.ietf.org/doc/rfc6282

[13] T. E. Winter, P. E. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," 2012.

[14] IETF CoRE, "Constrained RESTful Environments (core)," 2012. [Online]. Available: http://datatracker.ietf.org/wg/core

[15] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung, "MAC Essentials for Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 222–248, 2010.

[16] K. S. J. Pister and L. Doherty, "TSMP: Time synchronized mesh protocol," in *Proc. of Parallel and Distributed Computing Systems (PDCS)*, 2008.

[17] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, no. c, 2002, pp. 1567–1576.

[18] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004, pp. 95–107.

[19] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. Le Roux, "Poster abstract: WiseMAC, an ultra low power MAC protocol for the wiseNET wireless sensor network," in *Proc. of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003, p. 302.

[20] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proc. of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006, pp. 307–320.

[21] E.-Y. Lin, J. Rabaey, and A. Wolisz, "Power-efficient rendez-vous schemes for dense wireless sensor networks," in *Proc. of the IEEE International Conference on Communications (ICC)*, 2004, pp. 3769–3776 Vol.7.

[22] A. Warrier, M. Aia, and M. Sichitiu, "Z-MAC: A Hybrid MAC for Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 511–524, Jun. 2008.

[23] G.-S. Ahn, S. G. Hong, E. Miluzzo, A. T. Campbell, and F. Cuomo, "Funneling-MAC," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006, p. 293.

[24] J. Ko, A. Terzis, S. Dawson-Haggerty, D. Culler, J. Hui, and P. Levis, "Connecting low-power and lossy networks to the internet," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 96–101, Apr. 2011.

[25] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)," in *Proc. of the 7th*

IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Oct. 2011, pp. 365–372.

[26] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, Jul. 2005.

[27] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. E. Culler, and A. Terzis, "Evaluating the Performance of RPL and 6LoWPAN in TinyOS," in *Proc. of the Workshop on Extending the Internet to Low power and Lossy Networks (IP+SN)*, 2011.

[28] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009, p. 1.

[29] N. Tsiftes, J. Eriksson, and A. Dunkels, "Low-power wireless IPv6 routing with ContikiRPL," in *Proc. of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2010, pp. 406–407.

[30] T. Watteyne, A. Molinaro, M. G. Richichi, and M. Dohler, "From MANET To IETF ROLL Standardization: A Paradigm Shift in WSN Routing Protocols," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 688–707, 2011.

[31] O. Gaddour and A. Koubâa, "RPL in a nutshell: A survey," *Computer Networks*, vol. 56, no. 14, pp. 3163–3178, Sep. 2012.

[32] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California Irvine, 2000.

[33] E. Wilde, "Putting Things to REST," UC Berkeley School of Information, Tech. Rep., 2007.

[34] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, Mar. 2012.

[35] Z. Shelby, "Embedded web services," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 52–57, Dec. 2010.

[36] B. C. Villaverde, D. Pesch, R. De Paz Alberola, S. Fedor, and M. Boubekeur, "Constrained Application Protocol for Low Power Embedded Networks: A Survey," in *Proc. of the 6th IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Jul. 2012, pp. 702–707.

[37] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A Low-Power CoAP for Contiki," in *Proc. of the 8th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, Oct. 2011, pp. 855–860.

[38] A. Dunkels, L. Mottola, N. Tsiftes, F. Osterlind, J. Eriksson, and N. Finne, "The Announcement Layer: Beacon Coordination for the Sensornet Stack," *Wireless Sensor Networks*, vol. 6567, pp. 211–226, 2011.

[39] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker, and C. Görg, "Implementation of CoAP and its Application in Transport Logistics," in *Proc. of the Workshop on Extending the Internet to Low power and Lossy Networks (IP+SN)*, 2011.

[40] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "Evaluation of constrained application protocol for wireless sensor networks," in *Proc. of the 18th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, Oct. 2011, pp. 1–6.

[41] W. Colitti, K. Steenhaut, and N. De Caro, "Integrating Wireless Sensor Networks with the Web," in *Proc. of the Workshop on Extending the Internet to Low power and Lossy Networks (IP+SN)*, 2011.

[42] A. P. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi, "Web Services for the Internet of Things through CoAP and EXI," in *Proc. of the IEEE International Conference on Communications (ICC) Workshops*, Jun. 2011, pp. 1–6.

[43] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "REST Enabled Wireless Sensor Networks for Seamless Integration with Web Applications," in *Proc. of the 8th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, Oct. 2011, pp. 867–872.

[44] Apache Software Foundation, "CouchDB," 2011. [Online]. Available: http://couchdb.apache.org

[45] F. Barsi, A. A. Bertossi, C. Lavault, A. Navarra, C. M. Pinotti, S. Olariu, and V. Ravelomanana, "Efficient location training protocols for heterogeneous sensor and actor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 3, pp. 377–391, 2011.

[46] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 215–233, Sep. 2003.

[47] M. Di Francesco, S. K. Das, and G. Anastasi, "Data Collection in Wireless Sensor Networks with Mobile Elements: A Survey," *ACM Transactions on Sensor Networks*, vol. 8, no. 1, pp. 1–31, Aug. 2011.

[48] G. Ghidini, C. M. Pinotti, and S. K. Das, "A Semi-Distributed Localization Protocol for Wireless Sensor and Actor Networks," in *Proc. of the 8th IEEE International Conference on Pervasive Computing and Communications (PerCom) Workshops*, 2010, pp. 438–443.

[49] S. K. Das, G. Ghidini, A. Navarra, and C. M. Pinotti, "Localization and Scheduling Protocols for Actor-centric Sensor Networks," *Networks*, vol. 59, no. 3, pp. 299–319, May 2012.

[50] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad Hoc Networks*, vol. 2, no. 4, pp. 351–367, Oct. 2004.

[51] F. Barsi, A. A. Bertossi, F. Betti Sorbelli, R. Ciotti, S. Olariu, and C. M. Pinotti, "Asynchronous corona training protocols in wireless sensor and actor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 8, pp. 1216–1230, 2009.

[52] A. A. Bertossi, S. Olariu, and C. M. Pinotti, "Efficient corona training protocols for sensor networks," *Theoretical Computer Science*, vol. 402, no. 1, pp. 2–15, 2008.

[53] A. Navarra, C. M. Pinotti, V. Ravelomanana, F. Betti Sorbelli, and R. Ciotti, "Cooperative training for high density sensor and actor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 5, pp. 753–763, 2010.

[54] J. F. C. Kingman, *Poisson processes*. Oxford University Press, 1993.

[55] P. Guo, Q. Zhang, and T. Jiang, "Sleep scheduling in critical event monitoring with wireless sensor networks."

[56] B. Liu and D. Towsley, "A study of the coverage of large-scale sensor networks," in *Proc. of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2004, pp. 475–483.

[57] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge University Press, 1995.

[58] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, p. 32, 2007.

[59] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links," in *Proc. of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2009, pp. 133–144.

[60] G. Ghidini and S. K. Das, "An Energy-efficient Markov Chain-based Randomized Duty Cycling Scheme for Wireless Sensor Networks," in *Proc. of the 31st IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2011, pp. 67–76.

[61] ——, "Energy-efficient Markov Chain-based Duty Cycling Schemes for Greener Wireless Sensor Networks," *ACM Journal on Emerging Technologies in Computing (JETC)*, vol. 8, no. 4, Oct. 2012.

[62] Sun Labs, "Sun SPOT World," 2010. [Online]. Available: http://sunspotworld.com/

[63] S. Albers, "Energy-efficient algorithms," *Communications of the ACM*, vol. 53, no. 5, p. 86, May 2010.

[64] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proc. of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004, pp. 39–49.

[65] Sun Microsystems, "Sun Small Programmable Object Technology (Sun SPOT) Developer's Guide," 2009.

[66] ——, "Sun SPOT Theory of Operation," 2009.

[67] B. Bollobás, *Random Graphs*. Cambridge University Press, 1985.

[68] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 2003.

[69] A. Goel, S. Rai, and B. Krishnamachari, "Sharp thresholds for monotone properties in random geometric graphs," in *Proc. of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, 2004, pp. 580–586.

[70] G. S. Kasbekar, Y. Bejerano, and S. Sarkar, "Lifetime and coverage guarantees through distributed coordinate-free sensor activation," in *Proc. of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2009, pp. 169–180.

[71] Q. Cao, T. F. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *Proc. of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005, pp. 20–27.

[72] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003, pp. 171–180.

[73] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *Proc. of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006, pp. 321–334.

[74] S. Du, A. K. Saha, and D. B. Johnson, "RMAC: A routing-enhanced duty-cycle MAC protocol for wireless sensor networks," in *Proc. of the 26th IEEE International Conference on Computer Communications (INFOCOM)*, 2007, pp. 1478–1486.

[75] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "On Maximizing the Lifetime of Delay-Sensitive Wireless Sensor Networks with Anycast," in *Proc. of the 27th IEEE Conference on Computer Communications (INFOCOM)*, 2008, pp. 807–815.

[76] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *Proc. of the 5th ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2007, pp. 321—-334.

[77] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *Proc. of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2008, pp. 1–14.

[78] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proc. of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2010, pp. 1–14.

[79] F. Wang and J. Liu, "Duty-cycle-aware broadcast in wireless sensor networks," in *Proc. of the 28th IEEE Conference on Computer Communications (INFOCOM)*, 2009, pp. 468–476.

[80] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *Proc. of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2005, pp. 2470–2481.

[81] Y. Gu, T. Zhu, and T. He, "ESC: Energy Synchronized Communication in sustainable sensor networks," in *Proc. of the 17th IEEE International Conference on Network Protocols (ICNP)*, 2009, pp. 52–62.

[82] MEMSIC Corp., "Wireless Modules," 2010. [Online]. Available: http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html

[83] Mobile Integrated Solutions LLC, "MobiLinc App," 2011. [Online]. Available: http://www.mobileintegratedsolutions.com

[84] Demand Media Inc., "LiveStrong iPhone iPad App Calorie Tracker," 2011. [Online]. Available: http://www.livestrong.com/thedailyplate/iphone-calorie-tracker

[85] Nike, "Nike+," 2010. [Online]. Available: http://nikerunning.nike.com/

[86] Groundspeak Inc., "Geocaching Live," 2011. [Online]. Available: http://www.geocaching.com/live/default.aspx

[87] Foursquare Labs Inc., "foursquare," 2011. [Online]. Available: https://foursquare.com

[88] V. Gupta, P. Udupi, and A. Poursohi, "Early Lessons from Building Sensor.Network: an Open Data Exchange for the Web of Things," in *8th IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Mar. 2010, pp. 738–744.

[89] Apple, "Apple iPhone," 2010. [Online]. Available: http://www.apple.com/iphone/

[90] Google Inc., "Android," 2011. [Online]. Available: http://www.android.com/

[91] U.S. Government, "Data.gov," 2011. [Online]. Available: http://www.data.gov

[92] G. Ghidini, V. Gupta, and S. K. Das, "SNViz: Analysis-oriented Visualization for the Internet of Things," in *Urban Internet of Things Workshop*, 2010.

[93] Smart Meter Texas, "Smart Meter Texas," 2011. [Online]. Available: http://www.smartmetertexas.com

[94] Google Inc., "Google Nexus One," 2011. [Online]. Available: http://www.google.com/phone/detail/nexus-one

[95] MIT SENSEable City Lab, "The Copenhagen Wheel Project," 2010. [Online]. Available: http://senseable.mit.edu/copenhagenwheel

[96] UCLA's Center for Embedded Networked Sensing, "Biketastic," 2010. [Online]. Available: http://biketastic.com/

[97] G. Ghidini, S. K. Das, and V. Gupta, "FuseViz: A Framework for Web-based Data Fusion and Visualization in Smart Environments," in *Proc. of the 9th*

IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS), 2012.

[98] G. Ghidini and S. K. Das, "Improving Home Energy Efficiency with E2Home: A Web-based Application for Integrated Electricity Consumption and Contextual Information Visualization," in *Proc. of the 3rd IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2012.

[99] R. L. Ackoff, "From Data to Wisdom," *Journal of Applied Systems Analysis*, vol. 19, pp. 3–9, 1989.

[100] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. of the 6th Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[101] W3C, "Scalable Vector Graphics," 2011. [Online]. Available: http://www.w3.org/Graphics/SVG

[102] M. Bostock, "D3.js: Data-Driven Documents," 2011. [Online]. Available: http://mbostock.github.com/d3

[103] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, 2011.

[104] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava, "Biketastic: Sensing and Mapping for Better Biking," in *Proc. of the 28th International Conference on Human Factors in Computing Systems (CHI)*, 2010.

[105] Oracle Sun Labs, "Sensor.Network," 2010. [Online]. Available: http://sensor.network.com

[106] M. Bostock and J. Heer, "Protovis: a graphical toolkit for visualization." *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1121–1128, 2009.

[107] J. C. Anderson, J. Lenhardt, and N. Slater, *CouchDB: The Definitive Guide*, 1st ed. O'Reilly, 2010.

[108] W3C, "HTTP/1.1: Header Field Definitions," 1999. [Online]. Available: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

[109] Mozilla Developer Network, "Same origin policy for JavaScript," 2011. [Online]. Available: https://developer.mozilla.org/en/Same_origin_policy_for_JavaScript

[110] CouchApp.org, "Simple JavaScript Applications with CouchDB," 2011. [Online]. Available: http://couchapp.org

[111] OAuth Community, "OAuth Community Site," 2011. [Online]. Available: http://oauth.net

[112] Google Inc., "Google PowerMeter," 2011. [Online]. Available: http://www.google.com/powermeter

BIOGRAPHICAL STATEMENT

Giacomo Ghidini is a 5^{th}-year Ph.D. student under the supervision of Dr. Sajal K. Das in the Dept. of Computer Science and Engineering at UTA. He is a member of the Center for Research in Wireless Mobility and Networking (CReWMaN) at UTA. In 2010, he held a research assistant position at Oracle Sun Labs in Menlo Park, CA, under the supervision of Dr. Vipul Gupta, where he developed an analysis-oriented visualization system for Sensor.Network, a data storage and exchange platform for the Internet of Things.

Giacomo received his B. Comp. Eng. and M. Comp. Eng. degrees from the University of Bologna, Italy, in 2004 and 2008, respectively. He worked on his master's thesis during a 6-month visit at CReWMaN on a scholarship of the College of Engineering at the University of Bologna. In 2006, he was an exchange student at the University of Technology, Sydney, Australia, on a scholarship of the University of Bologna. During his undergraduate studies, he interned at Siemens AG in Munich, Germany, in 2000 and 2001, where he was a member of the Information and Communication Network Division under the supervision of Hr. Dr. Peter Hannss.

His current research interests include energy-efficient duty cycling in wireless sensor networks and the integration of wireless sensor networks into smart environments.