# Multichannel Distributed Coordination for Wireless Sensor Networks: Convergence Delay and Energy Consumption Aspects

Dujdow Buranapanichkit

Department of Electronic and Electrical Engineering
University College London

A thesis submitted for the degree of Doctor of Philosophy
In Electronic and Electrical Engineering
August, 2013

# Statement of Originality

I, Dujdow Buranapanichkit, confirm that this work presented in this thesis is my own work. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signed_____

Date_____

# Acknowledgement

Obtaining of a PhD has never been a simple thing. However I am the luckiest student to have Dr. Yiannis Andreopoulos as my supervisor during my PhD studies at UCL, to guide and advise me, support my research, teach me the methodology to solve research problems and most importantly help me with the mathematical aspects included in this thesis, which will be invaluable for my entire career. I wish and have to express my sincere gratitude to him.

Many thanks to the knowledgeable and friendly Academics of the Telecommunications and Information Processing Group: Dr. John Mitchell, Dr. Miguel Rio, and Prof. Izzat Darwazeh for their advice, understanding and encouragement in my research thesis. Also many thanks to Dr. Kenneth Tong for his support of my research.

I would like to thank all the friends helping my research: Dr. Fei Qin, Dr. Xuewu Dai and George Smart for their support on my work. I would also like to thank all the PhD members of my group coinciding with my time at UCL: Dr. Davide, Dr. Safa, Dr. BoWen, Oluyemi, Mohammad Ashraful, Haixia, Yu, George, Pedro, Spyridon, Ryan, Dr. Marcus and Dr. Manoj for the memorable times that we had together.

Beyond all the above people, I would like to thank my family for the support and encouragement during my PhD life. Lastly, I express my sincere gratitude to the Thai Government Scholarship (Science and Technology), funded by the Royal Thai Government, for financially sponsoring my PhD studies.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| APP | Application layer |
| BI | Beacon Interval |
| BO | Beacon Order |
| BRIEF | Binary Robust Independent Elementary Features |
| CAP | Contention Access Period |
| CCA | Clear Channel Assessment |
| CDF | Cumulative Density Function |
| CFP | Contention Free Period |
| CLT | Central Limit Theorem |
| CMOS | Complementary metal–oxide–semiconductor |
| CPU | Central Processing Unit |
| CSMA/CA | Carrier Sense Multiple Access with Collision Avoidance |
| DCT | Discrete Cosine Transform |
| DPCM | Differential pulse-code modulation |
| DESYNC | Desynchronization Algorithm of Degesys *et al* |
| DPM | Dynamic Power Management |
| DSP | Digital Signal Processing |
| DTFDMA | Distributed Time Frequency Division Multiple Access |
| EM-MAC | Efficient Multichannel MAC |
| FAST | Features from Accelerated Segment Test |
| FTSP | Flooding Time Synchronization Protocol |
| GTS | Guaranteed Time Slot |
| HAA | Hardware Abstraction Architecture |
| IBS | Interval Based Synchronization |
| IEEE | Institute of Electrical and Electronics Engineers |
| JPEG | Joint Photographic Experts Group |
| KB | Kilobyte |
| kbps | Kilobit Per Second |
| MAC | Medium Access Control |
| MB | Megabyte |

| | |
|---|---|
| OSI | Open System Interconnection |
| PAN | Personal Area Network |
| PCO | Pulse Coupled Oscillator |
| PDF | Probability Density Function |
| PHY | Physical Layer |
| RBS | Reference Broadcast Synchronization |
| RFA | Reachback Firefly Algorithm |
| RTOS | Real-time operating system |
| RV | Random variable |
| SD | Super frame |
| SEM | Standard Error of the Mean |
| SO | Super frame Order |
| SS | Steady State |
| STD | Standard Deviation |
| SYNC | Synchronization |
| TDMA | Time Division Multiple Access |
| TFDMA | Time Frequency Division Multiple Access |
| TPSN | Timing-sync Protocol for Sensor Networks |
| TSCH | Time Synchronized Channel Hopping |
| TSMP | Time Synchronized Mesh Protocol |
| VSN | Visual Sensor Networks |
| WSNs | Wireless Sensor Networks |

# Abstract

This thesis develops new approaches for distributed coordination of data-intensive communications between wireless sensor nodes. In particular, the topic of *synchronization*, and its dual primitive, *desynchronization* at the Medium Access Control (MAC) or the Application (APP) layer of the OSI stack, is studied in detail. In Chapters 1 and 2, the related literature on the problem of synchronization is overviewed and the main approaches for distributed (de)synchronization at the MAC or APP layers are analyzed, designed and implemented on IEEE802.15.4-enabled wireless sensor nodes.

Beyond the experimental validation of distributed (de)synchronization approaches, the three main contributions of this thesis, corresponding to the related publications found below, are:

- establishing for the first time the expected time for convergence to distributed time division multiple access (TDMA) operation under the two main desynchronization models proposed in the literature and validating the derived estimates via a real-world implementation (Chapter 3);
- proposing the extension of the main desynchronization models towards multi-hop and multi-channel operation; the latter is achieved by extending the concept of reactive listening to multi-frequency operation (Chapter 4 and 5).
- analyzing the energy consumption of the distributed TDMA approach under different transmission probability density functions (Chapter 6 and 7).

Conclusions and items for future work in relation to the proposals of this thesis are described in Chapter 8.

# Chapter 1

# Introduction

Wireless sensor networks (WSNs) have received significant attention from the research community for a number of years. One of the key aspects in the energy and bandwidth efficiency of WSNs is how to transmit data from sensors to base station(s), or from sensor to sensor without packet collisions at the physical (PHY) layer. This is because, if packets collide at the physical layer of standard PHY designs of WSNs, this means that all transmitters wasted energy for no benefit and the transmission opportunity was also wasted as no receiver can recover useful information.

A crucial element in any Medium Access Control (MAC) or Application (APP) layer protocol aiming to avoid packet collisions under data intensive WSN communications is an *efficient synchronization or desynchronization mechanism*. Within the context of MAC or APP-layer design of WSNs, synchronization can be seen as the mechanism to align transmission intervals between co-located sensors before deciding which sensor should use which interval. Desynchronization is the dual primitive of this, i.e. each sensor is obtaining its own transmission interval at the MAC or APP layer with (ideally) no overlap between the transmission intervals of co-located sensors. In WSN nodes with different clock characteristics, once synchronization or desynchronization is achieved, well-known low-complex scheduling (e.g. round-robin) can be used in order to send packets from one node to another without collisions. Thus, due to their importance, (de)synchronization approaches in WSNs have been a long-standing research problem, especially the aspects related to *distributed (de)synchronization*, i.e. (de)synchronization without the use of a central (master) coordinating node.

Interestingly, various distributed (de)synchronization mechanisms for WSNs have been inspired from mathematical biology and natural sciences [106][110][111]. For example, it has been known for some time that a group of

fireflies emit light flashes in a synchronous manner in order to attract their partner. In this work, we consider theory and development of biology-inspired distributed (de)synchronization in small- and medium-scale WSNs, with a special emphasis on desynchronization, which has been shown to be better suited to MAC and APP-layer protocol design than synchronization. Beyond the theoretical aspects of such (de)synchronization mechanisms, the thesis proposes practical WSN multichannel protocol designs based on them and validates their performance and energy efficiency under real-world transmission rates stemming from variable-rate data gathering and transmission in WSNs.

## 1.1 Aim and Scope

The standard IEEE802.15.4 MAC that was designed with a view towards WSN deployments supports a typical CSMA/CA mechanism for collision detection and avoidance at the MAC and PHY layers of WSNs. However, when considering the stringent energy and bandwidth constraints of WSNs, this mechanism is not sufficient as it does not ensure all self-inflicted (i.e. caused by nodes within the WSN) interference and packet collisions are avoided. Therefore WSNs need to have an effective protocol to manage how to transmit the data from many sensor nodes to the base station (or from sensor to sensor) without data collisions and with high bandwidth utilization. A collision-free time division multiple access (TDMA) or time-frequency division multiple access (TFDMA) protocol is the one of the best schemes for this purpose. Such mechanisms can be achieved in a centralized or decentralized manner. Because decentralized schemes do not depend on a single entity (i.e. the coordinator node), the decentralized-based TDMA will be the most interesting approach for our study. In order to achieve this in an energy-efficient and robust manner, in this thesis we shall follow biology-inspired proposals based on pulsed coupled oscillators. Two further advantages of such mechanisms are their inherent robustness to noise and their inherent capability to self-adjust in the presence of nodes joining or leaving the WSN.

## 1.2 Literature Review

Given that the proposals are all built on standard-compliant MAC and APP layers of modern WSN hardware, the first part of the literature review discusses the IEEE802.15.4 standard, which forms the basis of all experimental setups of this work. Subsequently, MAC and APP-layer oriented (de)synchronization proposals are reviewed, with a special emphasis on desynchronization that forms one of the main aspects of this thesis. Since desynchronization principles are used to propose a new multichannel MAC protocol for WSNs, previous work on multichannel coordination in WSNs in then reviewed. Finally, given that energy efficiency is one of the crucial aspects of WSN deployments, the last part of the literature review is devoted to research work on dynamic power management.

### 1.2.1 The IEEE 802.15.4 Standard

The MAC and PHY layers of IEEE 802.15.4 are specified for low-rate wireless personal area networks [33]. A widely-used implementation of the IEEE 802.15.4 protocol is provided in the nesC programming environment in TinyOS. Similarly some RTOS (real-time operating systems) such as the Contiki RTOS, have also supported the IEEE 802.15.4 over standard-compliant hardware. The supported functionalities of IEEE 802.15.4 MAC are [35]:

- Generating network beacons (coordinator node) and synchronizing to the beacons
- Supporting the personal area network association and disassociation
- Employing the CSMA/CA mechanism for channel access, supported by clear channel assessment (CCA) and channel frequency selection in PHY layer
- Handling the guaranteed time slot (GTS) allocation
- Providing a link control mechanism for increased connection reliability between two MAC devices.

This thesis is concerned with real-world WSN communications over existing hardware and protocol infrastructure. As such, we build algorithms at the MAC layer, with particular emphasis on backward compatibility to the standard

IEEE802.15.4 MAC. This allows for wide applicability of the thesis proposals to a variety of existing and forthcoming standard-compliant transceiver hardware in WSNs.

### 1.2.2 Synchronization Approaches

There are many proposals in research literature for synchronization (or desynchronization) in WSNs. However, most such proposals have been designed and tested via simulation and/or assume operation at the PHY layer [1]–[7], where beacon collisions can be resolved via dedicated physical-layer modes of operation. Moreover, when considering PHY-layer (de)synchronization, transmission delays and end-to-end response times are easier to determine than when considering the problem of (de)synchronization at the MAC or APP layer. This is because MAC and APP-layer (de)synchronization is based on packet transmissions (and not PHY-layer beaconing [8]–[10][13][15][19]), which add extra uncertainties with respect to the actual transmission and response times of nodes. However, their advantage is that they do not require changes at the PHY or MAC layers (beyond trivial modifications such as setting retransmission waiting times to near-zero at the MAC) and can thus be deployed in a variety of IEEE 802.15.4-compliant systems.

In this section, we highlight several schemes proposed in the literature, with emphasis on proposals that have been implemented on real hardware or have reached near-deployment phase for real-world WSNs, either at the PHY or MAC and APP layers of the OSI stack.

One of the most popular proposals is the flooding time synchronization protocol (FTSP) [1][124], which is aiming for centralized clock synchronization across all nodes of a WSN. The node with the least identifier is chosen to be the source of the "standard" time. This node periodically sends the synchronization message. The other nodes receive the message and calculate the shift and relative frequency with respect to the node that is the source of the standard time.

Within a similar context as the FTSP, several algorithms have been proposed to address the clock drift amongst different nodes [2]. Most algorithms fall in the

category of *drift-constrained clock models*. A popular approach is the interval-based synchronization (IBS) [3][125][126]. In IBS, while two nodes communicate, they update the upper and lower bounds and integrate the selection of the maximum lower and minimum upper bounds of time interval to derive a time estimator. As such, this method derives the time and drift constraints imposed by a certain implementation.

In order to derive a collision-free communication schedule, proposals for *time-triggered systems* form a global time base that is distributed amongst the nodes via distributed clock synchronization [4]. The reachback firefly algorithm (RFA) is proposed to provide such a common time base, based on pulse-coupled biological oscillators (PCOs) [4][112]. In PCOs with RFA, the clock synchronization messages have not been broadcast by all nodes at the same time to avoid message collisions in these broadcasts. Instead, the *reachback response* is defined, which sends the synchronization messages with a random offset [4]. This highlights one of the main problems of synchronization (in contrast to desynchronization), i.e. the need to provision for mechanisms avoiding message or beacon collisions at the PHY layer.

Another approach to the problem of synchronization is the timing-sync protocol for sensor networks (TPSN), which provides network-wide time synchronization in sensor networks [5][6]. In TPSN, the network is established as a hierarchical structure. The root of a network is the master node and other nodes synchronize their clocks to the master node. The technique is based on broadcasts from the master node, which is called the signal node, along the hierarchical structure. The broadcasts follow the reference broadcast synchronization (RBS) method [7]. Specifically, the signal node broadcasts the synchronizing beacon without timestamp to its neighbor. Each node uses its receiving time of that reference beacon to synchronize its own clock. Finally, other works [20][21] focus on achieving the accuracy of the time synchronization in wireless sensor network to obtain the clock synchronization [22][23].

More recently, there have been proposals to solve the dual problem to synchronization, i.e. *desynchronization* of wireless sensor nodes. The most prominent proposal is the DESYNC algorithm, which proposes a novel desynchronization and time division multiple access (TDMA) protocol [8]. In DESYNC, the sensor nodes work in a periodic schedule. Each node broadcasts (or "fires") a desynchronization-oriented message every period based on the previous and next firings from other nodes. Based on the received fire message broadcasts, each node adjusts its own fire message to be at the mid phase between the previous and next fire messages received. As a result, once the nodes have converged to desynchronization, each node's TDMA slot starts at its firing (zero phase) and ends at the firing of the next node and all TDMA slots are guaranteed to be non-overlapping.

Similar techniques to DESYNC were proposed in [9]. There, the desynchronization algorithm is based on the PCO concept. Each node has a clock which ticks for a single period, e.g. from time 0.0 to time 1.0. When node hears any message from another node outside a specified interval of time (called the refractory period), it updates its local clock to synchronize and transmit the message at the same time or at least within a small refractory period [9]. When the clock reaches 1.0, it resets to zero and a fire (or beacon) packet has been broadcast. The process is repeated indefinitely in this manner and it ensures desynchronization is achieved between the participating nodes regardless of their clock drift.

To summarize the literature review on synchronization, within existing approaches, all centralized synchronization approaches are based on the premise of maintaining (and synchronizing to) a centralized clock. As an alternative, distributed desynchronization is a new primitive that allows for equidistant channel access times for WSNs organized in a fully-connected (single-hop) network topology while avoid collisions of synchronization messages, leading to collision-free TDMA that does not require the presence of a coordinating node

[8][9][13][81][84][89][90]-[92]. All existing desynchronization approaches are based on the principle of *reactive listening*, where nodes periodically broadcast short packets (so-called “beacon”, “pulse”, or “fire” messages [8][9][13]) and then adjust their next broadcast time based on the reception of fire messages from the remaining nodes sharing the allocated spectrum. Equivalent distributed synchronization algorithms have also appeared recently [9].

#### 1.2.2.1. Recent Advances and Applications based on Desynchronization

Since the original formulation of desynchronization within the context of WSNs [4][8] several authors extended properties of its basic reactive listening primitive in a number of ways. Extensions towards multihop topologies have been proposed by low-complex graph theory methods [15][18], or via broadcast of only a limited number of beacon messages to the immediate neighbors [82][88]. The effects of node mobility in desynchronization were discussed in recent work [83]. A desynchronization method based only on carrier sense (under scenarios with limited reception capabilities or even under adversarial conditions) was also proposed recently [93]. Under the knowledge of the total number of nodes, it was shown that maintaining one node with fixed beaconing (i.e. an “anchored” node) [81] allows for faster convergence to TDMA. A modification of desynchronization to allow for “self-adjustment” of the firing order (instead of resetting its phase to zero) was proposed recently be Klinglmayr and Bettstetter [85]. Other works focused on modifications to the basic desynchronization to allow for TDMA with: low-complexity scheduling [36][87], unequal slot sizes [9][86], as well as scheduling under discrete resources (non-continuous time) [89]. Finally, in our recent work [34] we proposed a time-frequency extension of the desynchronization process in order to achieve increased bandwidth efficiency and allow for low-complex distributed coordination across the multiple channels supported by the IEEE 802.15.4 standard for WSN communications. Overall, it is evident that distributed desynchronization remains a very active field of study as it

can significantly assist a variety of problems in WSN design, such as interference avoidance, asynchronous transmission in multihop networks and low-complex multichannel coordination at the MAC layer. Since the latter aspect forms a part of this thesis, it is discussed in more detail in the following section.

### 1.2.3 Multichannel MAC Approaches

MAC protocols control and the Link/PHY layer access times in a WSN and as such they are essential in minimizing congestion and interference amongst competing nodes in the network. One of the most efficient ways to minimize congestion and interference is to use multiple channels. This has been recognized recently by the ratification of the coordinator-based time-synchronized channel hopping (TSCH) mode of the IEEE 802.15.4e-2012 standard [29][30][140][141].

Mo *et al*. categorize multichannel MAC protocols based on their principles of operation [63]. Some of the most energy-efficient protocols for multichannel MAC in WSNs are based on the *parallel rendezvous* approaches as proposed in McMAC [65], MMSN [64], etc, where multiple nodes simultaneously attempt to synchronize in different channels. The efficiency of such approaches stems from the achievement of quick synchronization, which incurs less collisions and idle listening time (both of which are known to be the sources of substantial energy expenditure in WSN-oriented transceivers). On the other hand, the *single rendezvous* MAC approaches use a dedicated control channel, where coordination of transmission and reception amongst the WSN nodes takes place. Some of the earliest single rendezvous MAC protocols via a dedicated control channel are DCA-PC [68] and DPC [69], which are important schemes for multichannel MAC coordination because they allow for high number of channels and large-size packets to be used. Other approaches along these lines include MMAC [66] and MAP [142]; however, such approaches require strict time synchronization before starting the multichannel coordination, which may be hard to achieve in practice.

Concerning parallel rendezvous protocols, the key aspect is to decrease the energy consumption when each node switches channels frequently to avoid

persistence interference on a single channel. One of the most promising works in this domain is the EM-MAC protocol for WSNs by Tang *et al*. [44]. Each node within EM-MAC attempts to predict its receiver's wake up time and transmitters and receivers are designed to rendezvous in each channel without the use of a centralized control channel. While EM-MAC has indeed demonstrated very low energy consumption, it achieves low bandwidth efficiency as most of the time the nodes are in stand-by mode. Other parallel rendezvous protocols, such as CAM-MAC [67], were proposed to enhance the performance significantly based on the usage of a dedicated control channel. CAM-MAC does not require clock synchronization and does not allow for frequent channel switching. Along similar lines, Pister and Doherty [32] designed the TSCH protocol for multi-hop WSNs using channel hopping with a centralized controller. TSCH has been shown to achieve high bandwidth efficiency with moderate energy consumption while minimizing collisions and avoiding persistent interference within a single channel. As such, it was included within the IEEE802.15.4e-2012 standard as an optional mode of operation and an open-source effort to support the deployment of TSCH within its standardized mode was started recently (openWSN project [140]).

### 1.2.4 Dynamic Power Management in WSNs

The basic idea of dynamic power management (DPM) is to improve the energy conservation capability of a system with minimal or no effect on its operation and performance. Evidently, the topic of DPM is vast and any attempt to summarize the entirety of the literature around it would be futile. Thus, in this section we only highlight the essentials of DPM within the context of WSNs in order to support our developments and validation of desynchronization-based protocols with energy-efficient operation.

Within a WSN, it is self-evident that each node should be set to sleep mode or idle state when there is nothing to do, in order to conserve energy [95]. Modern RTOS designs, such as Contiki, support DPM inherently by automatically setting the transceiver or the processor (or both) to sleep mode when the RTOS detects

they are not used by the executing threads. On the other hand, other operating systems like TinyOS require the programmer to manually set parts of the sensor to sleep mode when they are not used.

As discussed previously, several MAC protocols are specifically attempting to minimize energy consumption. Overall, the crucial aspects to address are [70][73][77][95]–[98]: *(i)* minimizing idle listening, i.e. minimizing the times when a node switches on its receiver circuit when there is no packet to be received; *(ii)* minimizing the amount of collisions occurring in the wireless medium; *(iii)* minimizing the control operations required to achieve minimal (or no) idle listening and minimal (or no) collisions, as very complex MAC protocols will not allow for the processor to go to sleep mode. Amongst the vast amount of research works in this area, we highlight two representative cases that encompass most or all of the principles highlighted previously: Pantazis *et al* [96] proposed a TDMA-based scheduling using sleep mode to save energy consumption; the monitoring of an electric system [97] utilized WSNs with DPM to allow for very long operational times. The reader is referred to the survey paper of Bachir *et al* [98] for further examples of DPM-based approaches in WSNs.

Another category of research approaches the problem of DPM by focusing on a particular aspect of the WSN-based monitoring [70]. Technology-oriented approaches design new circuits and systems for more efficient energy management [71][72], or strive for more efficient scheduling and transmission protocols [34][44][47][58]. These try to bridge the gap between data sensing and transmission requirements and the corresponding energy production (e.g. via a harvesting unit) and energy storage capability of the underlying hardware. Finally, another group of approaches proposes optimal energy management policies under given energy harvesting, sensing and transmission capabilities [73]-[75][78]. Such policies optimize the manner each sensor node performs its data gathering and buffer management in order to minimize the required energy consumption.

## 1.3 Thesis Structure

The vast majority of these works do not provide a practical implementation; instead, only simulation results are presented. Importantly, there is no common experimental basis to compare the algorithms within each category of Figure 1.1. Performing detailed theoretical and experimental comparisons between distributed (de)synchronization algorithms based on local clock information is the first aspect of this thesis.

To this end, the main objective of this thesis is to establish, analyze and evaluate the distributed synchronization protocols operating on real WSN hardware. The major contributions are outlined below:

(i) We propose realizations of the (de)synchronization algorithms that are simple enough to run on commodity WSN hardware and the results are measured from a practical test-bed using IEEE802.15.4 MAC [19][33].

(ii) We propose (for the first time) an analytic model for the convergence to desynchronization under PCO-based methods, which is coupled with experimental results using real WSN hardware based on the IEEE 802.15.4.

(iii) The bandwidth efficiency of desynchronization is enhanced by introducing desynchronization in multiple communication channels available under the IEEE802.15.4 MAC [33] and also by considering desynchronization under a multi-hop topology.

(iv) We prove the convergence of the proposed distributed multichannel desynchronization and propose a model for the expected delay to achieve convergence to steady-state; the theoretical results are validated via experiments based on a nesC TinyOS deployment [34].

(v) The energy efficiency for the distributed synchronization in WSNs is studied according to a model for energy consumption under a variety of statistical characterizations for the data production [77].

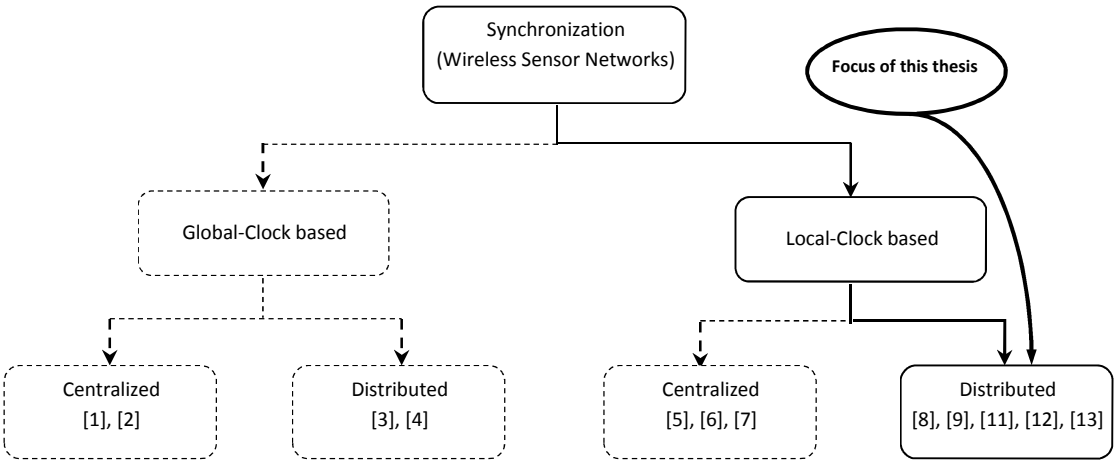


Figure 1.1: Schemes for time synchronization in wireless sensor networks.

Concerning (i) and (ii), for the two main proposals of PCO-based desynchronization:

- the DESYNC-based TDMA protocol was implemented on TinyOS-based Telos sensor motes [8];
- the decentralized PCO-based synchronization algorithm of [11] was implemented on TinyOS-based MicaZ motes.

In order to evaluate and compare between these two methods, our work designs and implements both algorithms on the same hardware, i.e. TinyOS-based iMote2 nodes.

Concerning (iii)–(iv), we propose two elements for improving the network performance of PCO-based desynchronization protocols. In existing PCO-based synchronization primitives, besides the time dimension, frequency (i.e. the utilized channel of IEEE 802.15.4) is not considered. To this end, we design protocols for adjusting the transmission channel based on the number of nodes already present in each channel. Via experiments with real sensor network hardware (iMote2 with TinyOS), we demonstrate that the proposed mechanisms lead to significantly increased bandwidth efficiency in comparison to [8][11] as well as improved convergence time in multichannel desynchronization compared by [32][44].

Finally, concerning (v), our contribution is to consider the expected energy consumption of a multichannel desynchronization-based protocol and sensor nodes producing and transmitting data with variable rates. The aim is to predict how much energy must be preserved under duty-cycling–based operation and under the existence of a predefined energy source (e.g. energy available from the battery within a certain interval of time). Specifically, we propose an analytic model that has been validated via different transmission rate probability density functions (PDFs) stemming from real-world applications. The average energy consumption was experimentally measured based on a resistor connected in series to TelosB motes (running the Contiki RTOS and the proposed multichannel desynchronization) and a high-speed oscilloscope capturing the current consumption in real time.

## 1.4 Publications

- H. Besbes, G. Smart, D. Buranapanichkit, C. Kloukinas and Y. Andreopoulos, "Analytic conditions for energy neutrality in uniformly-formed wireless sensor networks," *IEEE Transactions on Wireless Communications*, - Submitted.

- A. Vittorioso, D. Buranapanichkit, G. Fortino and Y. Andreopoulos, "Coordination for TDMA operation in WSNs: comparison between centralized and distributed mechanisms," *9th European Conference on Wireless Sensor Networks (EWSN)*, poster presentation, 2012

- D. Buranapanichkit, and Y. Andreopoulos, "Distributed time-frequency division multiple access protocol for wireless sensor networks," *IEEE Wireless Comm. Letters*, vol. 1, no. 5, pp. 440-443, Oct 2012.

- D. Buranapanichkit, A. Vittorioso, G. Fortino and Y. Andreopoulos, "Performance comparison of centralized and distributed coordination for

TDMA operation in wireless sensor networks," *London Communication Symposium (LCS)*, 2011.

# Chapter 2

# Synchronization Primitives

Synchronization in WSNs is a long-standing research problem [10][11][112][115]-[117]. In the last 10 years, new approaches based on pulse-coupled oscillators have gained significant traction within the WSN research community [8][9]. The basic premise of these methods is reactive listening: each wireless sensor node listens for periodically-broadcast messages from other nodes, called “pulse” or “fire” messages. Based on the received messages, each node adapts their internal time measurement function that triggers their own fire message [8][9]. Once the node is triggered to fire, it reacts by broadcasting its own fire message. The advantage of these protocols in comparison to centralized clock synchronization is in their implementation simplicity, robustness to clock drift and transmission delay jitter, and in the avoidance of depending on a single coordinator node (or base station).

The reactive listening process consists of a convergence period, where nodes adjust their firing times, and a steady-state period where fire messages are sent by each node in regular (periodic) intervals, followed by data packets. In the latter case, nodes have converged into a collision-free time-division multiple access (TDMA) system. It is well-known that convergence to TDMA can be achieved by synchronization [4][9] or by its dual primitive, i.e. desynchronization [8][15][18]. Our work studies both schemes. However the bulk of this thesis work focuses on the latter since: *(i)* no collision-avoidance schemes for the fire messages themselves are required (i.e. unlike in synchronization schemes [4]) and *(ii)* desynchronization algorithms have low complexity and their implementation in a WSN requires a single timer in each node [8][18].

Given the similar mechanisms used by all algorithms, in this thesis we use common notations for all distributed synchronization designs. They are

represented in the network diagram of Figure 2.1 where we depict the model of a fully-connected wireless sensor network. A broadcast message from a node is received from every node in the network. For non-invasive monitoring purposes, we assume there always exists a "base station" node that listens to all broadcast messages from all nodes and then sends the message trace file to the computer via the USB port for analysis.

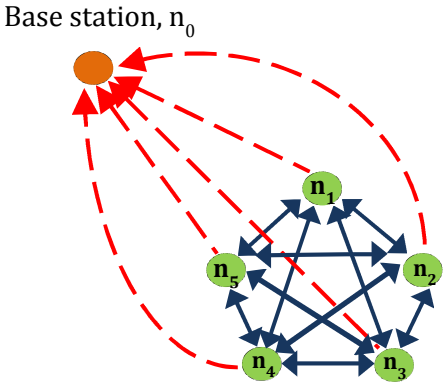


Figure 2.1: Network model for WSNs assuming 5 fully-connected nodes and one monitoring base station node (in the orange circle).

Each node $n_i$, $1 \leq i \leq W$, with $W$ being the total number of nodes in the network, picks its local time, $t_i^{(k-1)}$, to broadcast its $(k-1)$th fire message. The "local" view of the network events seen from node $n_i$ during its $(k-1)$th period is represented in Figure 2.2, where the received fire messages from other nodes are represented by downward arrows. Node $n_i$ broadcasts its fire message at $t_i^{(k-1)}$, which is indicated by the upward arrow. Throughout the thesis, $t_i^{(k-1)}$ comprises the time instant when node $n_i$ transmitted the fire message within the $(k-1)$th period (or $(k-1)$th firing), while $t_{i-1}^{(k-1)}$ and $t_{i+1}^{(k-1)}$ comprise fire messages received by node $n_i$ and originating by nodes $n_{i-1}$ and $n_{i+1}$ (respectively) during $(k-1)$th period. Without loss of generalization, we assume that nodes fire in their numerical order, i.e. node $n_{i-1}$ fires before node $n_i$ for each periodic interval. Figure 2.2 shows that node $n_i$ receives the broadcast fire message from the previous node at $t_{i-1}^{(k-1)}$ as well as from the next node at $t_{i+1}^{(k-1)}$. Based on the reactive listening primitive of each synchronization scheme, each node $n_i$ calculates the time to broadcast its own message during the next period $(k)$, i.e.

$t_i^{(k)}$. In the steady state of the WSN system, each node fires its message every $T$s, which is the desired TDMA period. The aim of each synchronization approach is to schedule all fire messages in a way that leads to TDMA, with each of the $W$ nodes having transmission interval of $\frac{T}{W}$s in steady state (SS), after $k_{\text{ss}}$ periods. The convergence to TDMA is checked by: $\left|t^{(k_{\text{ss}})} - t^{(k_{\text{ss}}-1)} - T\right| < b_{\text{thres}}T$, with $b_{\text{thres}} \epsilon$ [0.001,0.020] a preset threshold. Once this condition is satisfied, the system is deemed to be in a "converged" state and $k_{\text{ss}}$ comprises the number of the required firing cycles for convergence.

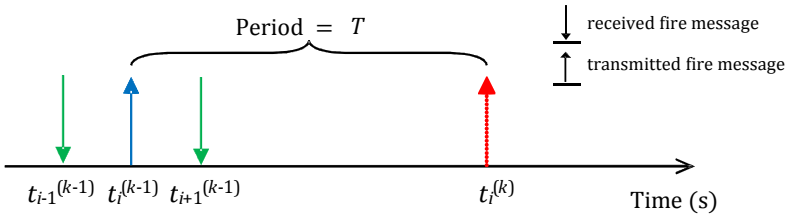


Figure 2.2: Local view of node $n_i$'s events including transmitted and received fire messages.

In order to achieve convergence to TDMA, beyond the default firing period of $T$ seconds, each distributed synchronization protocol is using a "coupling" mechanism during the reactive listening primitive with coupling parameters:

- $\alpha \in (0,1)$, which is the phase coupling constant controlling the speed of the phase adaptation; alternatively, $\alpha$ is called the "jump-size" parameter in [8]; this parameter is used in Sections 2.2 and 2.3;
- $\varepsilon > 0$, which is the coupling strength, used for tuning the scheduling in Section 2.1;
- $\varphi \in [0,1)$, which represents the node's phase variable. Each node has a local clock whose counter loops from 0 to 1: the phase variable represents a mapping of the node's clock value (modulo $T$) to the interval [0,1); importantly, this mapping may not always be linear or indeed continuous, as explained in the following section.

In this chapter, we describe the design and implementation of each primitive of the distributed (de)synchronization algorithms studied. First of all, the operation of the synchronization approach is presented, a protocol that we called the SYNC

algorithm [4]. This was inspired by firefly synchronization [4]. The dual primitive is the desynchronization scheme, which is called the DESYNC algorithm [8]. Finally, the description of the PCO-based algorithm with inhibitory coupling [9] is given, which is also a desynchronization approach. The chapter concludes with a discussion of all (de)synchronization primitives.

## 2.1 Synchronization

This section studies the distributed synchronization strategy. We implemented the SYNC algorithm based on the PCO and the RFA algorithms [4]. In the PCO scheme, each node needs to fire and receive the signal pulse. Similarly, each node has a clock that counts periodically from 0 to $T$ [9], i.e. $\varphi \in [0,1)$ . When the node's phase reaches 1, the node broadcasts its synchronization (fire) message and resets its counting clock (phase) to zero. When the other nodes hear this broadcast, they adjust their local clock according to the *coupling strength* ($\varepsilon$) and a concave down function [4][9][12]. Since each node $n_i$ uses a local state variable $x$ ($0 \leq x \leq 1$) to measure time within each period; the non-linearity of the concave down function used in existing proposals [4][9][12] makes the evolution of the phase variable a nonlinear function as shown Figure 2.3. Importantly, within each period of each node, the SYNC algorithm defines a special time interval, called the *refractory period*, which is defined at the beginning and the end of the time ($x$) measurement of each period. If a broadcast message is heard during this interval, the node is not allowed to update its clock. In fact, if all nodes are synchronized, they transmit the fire message at the same time, or within this refractory period, and no further phase update takes place.

The RFA is used in conjunction with PCO to enable a decentralized synchronization algorithm [11]. The reachback firefly algorithm helps where nodes purse synchronization by adapting the offset in order to avoid the problem that a sender cannot receive messages while it transmits the data. In the original PCO proposal, when all nodes are synchronized, they will transmit the clock message at the same time using a special medium access mechanism [9]. However,

in conventional MAC designs for WSNs, e.g. in IEEE802.15.4 MAC, this causes collision of the fire messages themselves. For this reason, each node should send the synchronization (fire) message with a random offset to avoid collisions. Nodes collect all synchronization events from the last period and decide how to react during the next period. The messages themselves contain the relative time from the moment they should have been fired, to the moment they were actually fired (because of the random offset). This is called the reachback response [4].

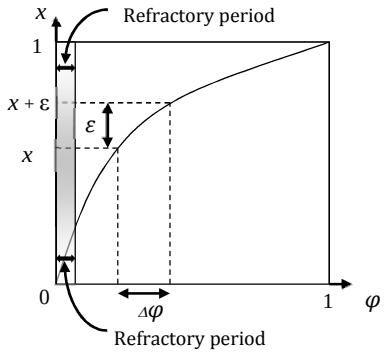


Figure 2.3: PCO Dynamics model demonstrating the mapping between the state variable $x$ counting the clock period and the node's phase $\varphi$.

In our work, we designed and implemented these approaches for the SYNC algorithm as explained in this next subsection.

### 2.1.1 Implementation Details

The description of the design and implementation of the SYNC algorithm for IEEE802.15.4 MAC is divided in two sections. The first subsection presents the SYNC algorithm during the convergence period. The second subsection is concerned with the operation after convergence and introduces the TDMA protocol and its operation in IEEE802.15.4 MAC.

An important characteristic of the PCO with RFA is the distinction between the notions of *algorithmic time* and *physical time* as shown in Figure 2.4. The algorithmic time represents the fire message broadcast within the refractory period, which will happen when the system is in converged state. This is the time used in the PCO-based synchronization algorithm for establishing convergence to steady (synchronized) state. By properly choosing the refractory period, the system can be stabilized as demonstrated in [121][122]. The physical time is the

fire message broadcast time assigned based on the reachback response in order to avoid fire message collisions. This is the actual (physical) time that the node will broadcast the fire message. Therefore, we have to identify the delta time ($\Delta t$) which is the difference between the physical time and the algorithmic time. This variable needs to be placed in the fire message itself. In this way, once a node receives (or broadcasts) the fire message, it (or all other nodes) will be able to calculate and consider the algorithmic time to check for convergence to TDMA with the SYNC algorithm.

In order to calculate the physical time for sending the fire message, we need to assign the slot number for each node. The physical time was set to be the time at the slot number timing with $\frac{T}{W}$ s. This time was also used for starting sending the data messages when nodes are synchronized.

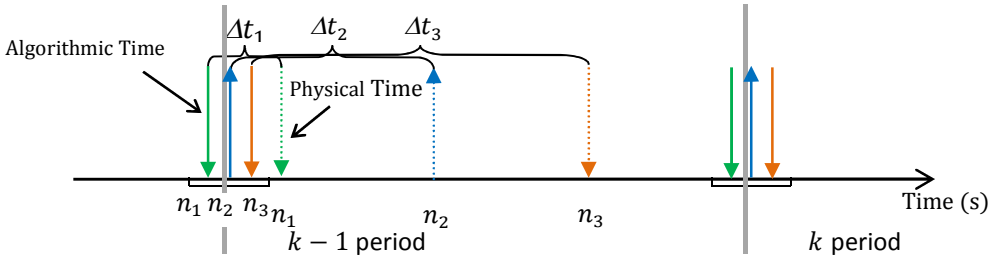


Figure 2.4: Local view of decentralized synchronization algorithm indicating the algorithmic fire times (in solid lines) and the physical fire times (in dotted lines).

### 2.1.1.1 Operation of PCO-based synchronization during the convergence period

As shown in Figure 2.5, we consider only the algorithmic time when checking for synchronization. Each node $n_i$ calculates its phase, $\varphi_i^{(k)}$, which is the difference between the next fire time and the received fire time from the other nodes. As mentioned, the phase is an important variable in the SYNC algorithm as it is used to control the time $t_i^{(k)}$ to broadcast the node's own fire message during the $k$th periodic interval. After node $n_i$ broadcasts its fire message, it will calculate the time to schedule $t_i^{(k)}$, i.e. its next fire time. During the convergence period of PCO with RFA, each node keeps all received fire messages from the other nodes in the previous period $(k-2)$ as demonstrated in Figure 2.6. They are collected in the node's *event list*, which is a small buffer in memory. Using the event list buffer, each

node will use the algorithmic time of all received fire message in period $k-2$ and assume they happen in the same manner in $k-1$. Based on this assumption, the node will calculate $\varphi_i^{(k)}$ and decide when its next fire broadcast will take place in the next period ($k$).

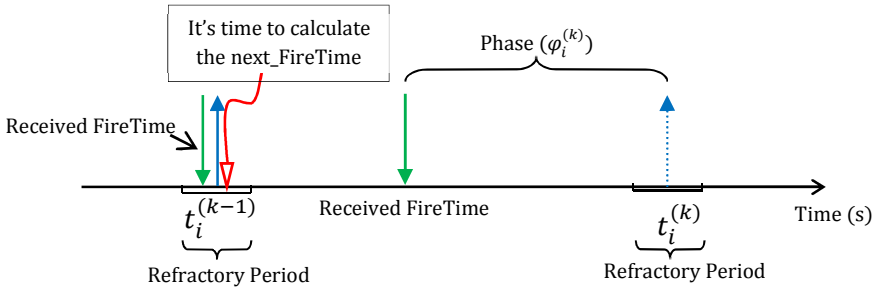


Figure 2.5: General view of local timeline for SYNC algorithm.

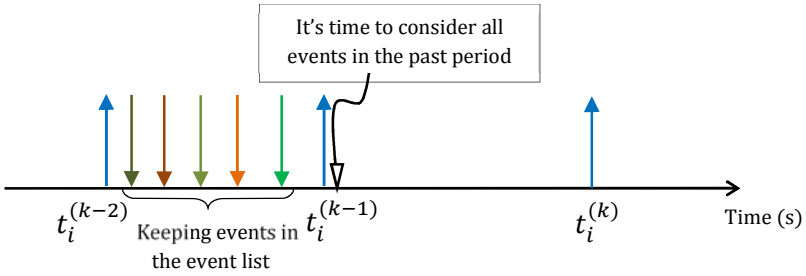


Figure 2.6: Local timeline for SYNC algorithm with buffer.

Concerning the format of a fire message: Each node will store $\Delta t$ (difference between physical and algorithmic time) in 1 byte. As shown in Figure 2.7, each fire message also includes the node ID, the message sequence number and the delta time. Since the format for storing time variables in TinyOS has 4 bytes, $\Delta t$ is converted to 1 byte. The value of $\Delta t$ will be divided by a time constant (12800) to fit within 1 byte.

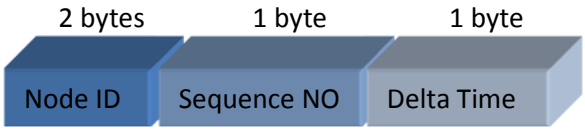


Figure 2.7: Format of a fire message of SYNC algorithm.

We now start to describe our TinyOS implementation from the event of receiving a fire message, which is shown in Figure C.1 in Appendix C. Each node will determine the algorithmic time from the obtained fire message (line 3). Thus, it also stores that algorithmic time plus one period (since it will be used for the

next period) in the event list. When it is time to fire its own message, the node will check first that it has already computed its algorithmic fire time. In fact, if *just_usedAlgTime* is FALSE, the node will need to calculate its own algorithmic fire time. To this end, it will start to process all received fire times from all other nodes in order to set the new next algorithmic fire time. After finishing the execution, it will clear all the received algorithmic time in event list in order to keep the new algorithmic times to be received in the next period. Furthermore, it will set its physical time at the start interval of its slot time (line 10). After that, the case of *just_usedAlgTime* = TRUE is handled; in this case, the node will send its broadcast fire message at the calculated physical time (line 12). Node then will reset its algorithm time (line 13).

When setting the physical time, we also calculate the slot time for each node, i.e. the time interval during which the node can transmit data. The value controlling the slot time is the *slot number*, which is found by checking the received fire message as described in the following. As shown in the code of SetPhysicalTime{} of Figure C.1, the *slotTime_MyNode* is the start of the slot time. It is also used to assign the physical time for each node. In addition, the *slotTime_EndNode* is the end time of the node's (transmission) slot. It is imperative to set the slot time correctly to ensure collision-free transmissions during each node's slot. Because of noise in the measured times of the start and end period, a small time value is subtracted from the *slotTime_EndNode* to avoid colliding with the next received fire time.

Consider determining $t_i^{(k)}$ of a certain node, as shown in the schematic of Figure 2.8. All received fire times were sorted from first to last and the node will check these fire times. If the calculated algorithmic time of a received fire message is within the refractory period, the node will not change its fire time (i.e. this message is ignored). Otherwise, $t_i^{(k)}$ is adjusted according to the time difference between the expected next fire time ($t_i'^{(k)} = t_i^{(k-1)} + T$ is the expected next fire time) and the received algorithmic fire time, which we call the delta-phase ($\Delta\varphi$). When

$\Delta\varphi > 0.5$, we consider the received fire time in " phase A", which means its received fire time is closer to $t_i^{(k-1)}$ than $t_i'^{(k)}$. Conversely, when $\Delta\varphi \leq 0.5$, this means the received algorithmic fire message time is closer to $t_i'^{(k)}$ than $t_i^{(k-1)}$. We consider the received fire time in "phase B".

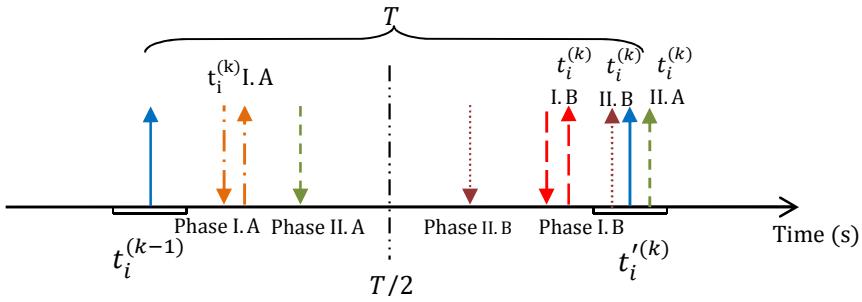


Figure 2.8: Local view for determining $t_i^{(k)}$ of SYNC algorithm.

The node will schedule its own algorithmic fire time based on the difference between the currently-scheduled firing time and the received algorithmic fire time. If these times are close to each other, the node will fire instantly when it gets that fire message. In the case where the received fire time is more than $T/2$ from the node's own fire time, the node will adjust $t_i'^{(k)}$ by adding or subtracting by a constant value $\varepsilon$ to obtain the new $t_i^{(k)}$. In the PCO-based protocol the appropriate value of the coupling strength [119][120], $\varepsilon$, was within an interval of a few milliseconds in order to reach quick and stable synchronization [9].

As shown in the pseudocode of Figure C.2 in Appendix C if $\Delta\varphi$ is less than the refractory period, the *just_Phase0* will be TRUE and $t_i^{(k)}$ will be scheduled at the normal period (i.e. after $T$s). Meanwhile, if $\Delta\varphi$ is out of the refractory period, the node will adjust it as described previously: if $\Delta\varphi > 0.5$, the node will add $\varepsilon$ to its phase (line 10). Afterwards, the new phase will be checked again to see if it exceeds unity (one period). If so, the node will fire after a very small offset value (line12). Otherwise, $\varphi_i^{(k)}$ is simply adjusted by $\varepsilon$ (line 14). Conversely, if $\Delta\varphi \leq 0.5$, node will subtract $\varepsilon$ (line 16). Then, if the new phase is less than 0, node will fire instantly (within a small offset value – line 18). Otherwise it will fire in the new time scheduled after the subtraction of $\varepsilon$ (line 20).

#### 2.1.1.2 Operation of PCO-based synchronization after convergence is achieved

The data transmission in the SYNC algorithm is shown in Figure 2.9. Each node starts sending data after checking for convergence to TDMA. Convergence is established when all the algorithmic times from all nodes are in the refractory period; when this is satisfied per node, each node assumes that it is in converged mode. After this point, the node will transmit data messages to a receiver node after broadcasting its fire message and within its calculated transmission slot time. These data messages are from the data in the flash memory of the TinyOS-based mote. When the node has already sent a packet, it will check the remaining time of its slot time. If the rest of time is less than the *time_SendData_onePack* (Figure 2.9), the node will stop sending data.

Besides transmitting and receiving data, each node has to write the received data in its flash memory. When each node receives data messages, it keeps them in the "received" buffer. Afterwards, once the node acquires a fire message from another node, it will write all its data message to the flash memory. This method ensures all messages are written correctly. Each node sends its data messages consecutively during its slot time.

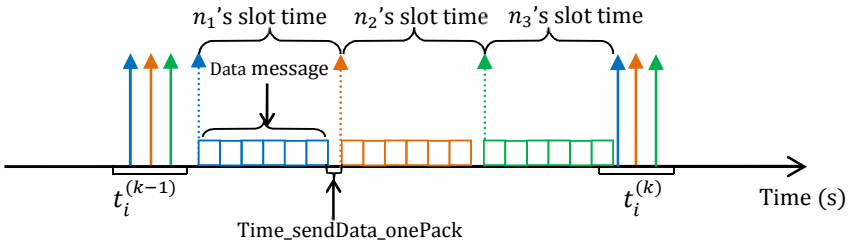


Figure 2.9: Local view of SYNC algorithm when transmitting data.

For the format of the data message, we changed the size of the data length for iMote2 wireless sensors at path *beta/platform/imote2/AM.h*. The TOSH_DATA_LENGTH was modified to 28. This data message includes the source mote ID, sequence number and data which were read from the flash memory of the node; the format of the data message is shown in Figure 2.10.

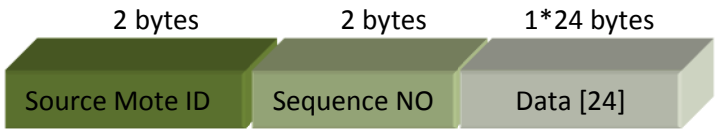


Figure 2.10: Format of a data message of SYNC algorithm.

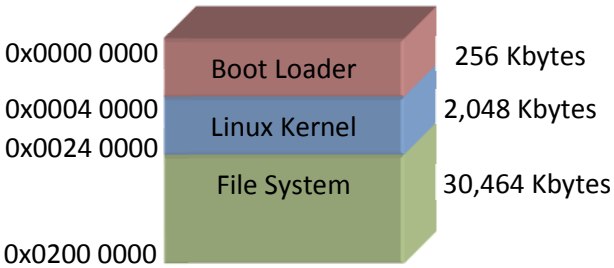


Figure 2.11: Address space of the flash memory of iMote2 [123].

The iMote2 sensor node has 32 MB of flash memory, shown schematically in the memory address space of Figure 2.11. Each node has the file system in the flash memory and the space to store all outgoing data starts at address 0x0030 0000. Correspondingly, all incoming (received) data is written in memory positions starting from address 0x0040 0000

As mentioned, before starting to send data, each node checks for convergence. Moreover, while each node sends its fire message, it will check the number of neighbour nodes, which is counted based on the number of received fire message broadcasts. At the same time, it will count the number of fire messages received within the refractory period, which is called "phase0" in the pseudocode of Figure C.3 in Appendix C. As shown in the pseudocode, if the number of neighbour nodes and the number of algorithmic phases are equal and also *just_Phase0* = TRUE (line 15), the node will be in converged state. As a result, the node transmits data messages to a receiver node after it broadcasts a fire message. Those data messages are from the data in flash memory of the TinyOS-based mote. When the node has already sent the packet out, it will check the remaining time of its slot time. If the rest of time is less than the *time_SendData_onePack*, the node will stop sending data. The last activity of a node is to write data in flash memory when it already has the data message in the received buffer. This is done after the node receives a fire message from another node.

### 2.1.2 Experimental Validation

The SYNC algorithm was implemented on iMote2 wireless sensor motes. The motes use the standard (Zigbee) IEEE802.15.4 protocol theoretically capable of 250kbps

peak rate based on the 2.4GHz CC2420 RF transceiver. Radio messages use TinyOS's standard and are conventionally called TOS messages. The active message format consists of a 28-byte payload including the sender ID and message sequence number and a 12-byte header including the message length and receiver ID.

We use the TinyOS's implementation for the transmission with the CSMA radio interface. In our experiment, we set the initial backoff to approximately 1.2 ms for data sending. We changed the value of initial backoff from the preset of a random number as "*(call Random.rand () & 0xF) + 1*" to a fixed value ("4") at a library file in the path *tos/lib/CC2420Radio/CC2420RadioM.nc*. This number becomes to be equal to 4*20 symbols*16μs per symbol = 1.28 ms. This is because, under the SYNC protocol, collisions are expected to happen very rarely.

#### 2.1.1.1. Wireless Sensors and their hardware and software components

The concept of a sensor in this thesis comprises a hardware platform with a transceiver. Figure 2.12 shows an example platform, the iMote2 from Crossbow [60], which is designed for advanced sensor network applications requiring high CPU/DSP and reliability. It has an Intel PXA271 32-bit microcontroller with 256 KB of SRAM and 32 MB of SDRAM and 32 MB of flash memory. Its radio unit is a TI CC2420 transceiver with an on-board antenna that follows the IEEE802.15.4 standard (Section 1.2.1). The CC2420 supports a 250 kbps data rate at the PHY layer with 16 channels in the 2.4 GHz band [17]. The iMote2 is a modular stackable platform and can be stacked with sensor boards to customize the system to a specific application, along with a battery board to supply power to the system.

The other node (mote) used in the results of this thesis (Chapter 6 and 7 with Contiki), the Telos [61], is designed for very low-power operation and can support less complex tasks and applications than iMote2. It has a TI MSP430 16-bit microcontroller with 10 KB of RAM and 48 KB of flash (program) memory, uses the same CC2420 radio chip, and also operates with a pair of AA batteries.

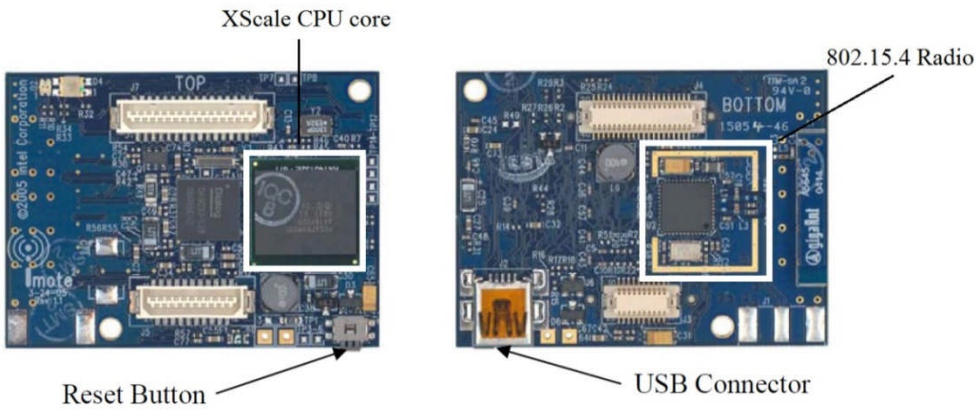


Figure 2.12: Photos of the iMote2 board

Concerning operating systems used in such WSN hardware, the first mainstream operating system in use today is TinyOS. TinyOS is a lightweight operating system specifically designed for low-power wireless sensors [143]. TinyOS applications and systems are written in the nesC language which is similar to C language but produces executable modules with reduced code size. TinyOS is an event-driven operating system using a component-oriented programming abstraction. For example of a basic data-aggregation application, When TinyOS tells the application that the node has completed the boot process, the application code configures the power settings on the radio and starts a periodic timer. Depending on the specifics of the functionality of each application, this timer fires every few milliseconds (for example) and the application code puts the sensor values into a packet and calls the radio to send the packet to a data sink. TinyOS can be used to port applications easily to a variety of hardware platform via a multi-level hardware abstraction architecture (HAA) which provides access to a device i.e. radio, timers, sensor, etc. as PHY-layer components. In addition, when the nodes organize themselves into a network, TinyOS provides MAC-layer implementations for communications-oriented applications.

The other operating system gaining significant abstraction for sensor network applications is the Contiki operating system. Contiki is an open source operating system for networked systems focusing on low-power wireless devices. Contiki provides multitasking and a built-in TCP/IP stack. The Contiki programming model

is based on a memory-efficient programming abstraction with multi-threading and event-driven programming. This feature is quite similar to nesC TinyOS for implementing applications.

#### 2.1.1.2. Experimental Setup

We formed a single-hop network composed of up to 16 TinyOS motes. One mote is set up as a base station, i.e. it records all the messages (fire messages and data messages) over the USB port to a computer. The base station has designated start and end time so as to keep all messages within one minute (60 s). We used the same fixed parameters which are:

- period ($T$) set to 2 seconds,
- epsilon ($\varepsilon$) set to 20 ms and refractory period set to 60 ms (i.e. ±30 ms before and after the node's own fire time)

for all experiments. The number of motes was selected as: 4, 8, 12 and 16. Five tests of 60s for each number of motes were performed.

In this section, we study the performance of all distributed TDMA i.e. SYNC (in subsection 2.1.1.3), DESYNC (in subsection 2.2.2.2) and PCO-DESYNC (in subsection 2.3.2) to compare the bandwidth efficiency with the same hardware platform and settings. At the beginning, in order to measure the maximum data rate that can obtain at the APP layer with IEEE 802.15.4–enabled sensor nodes, we deployed one node to transmit messages to the base station without interruption or other concurrent transmissions. We then measured the APP-layer data throughput at the base station, which was found to be 84.8 kbps. This is used as our benchmark in all our comparisons based on the concept of the *normalized* (%) *throughput*, which defines the percentile ratio between the throughput of an experiment (e.g. testing the Sync protocol) and the maximum measured throughput of 84.8 kbps. Beyond the normalized throughput, the *max* and *min individual throughputs* were calculated from best and worst average throughput per individual node. Finally, for data messages, given that we include the source mote ID and sequence number,

we can evaluate the *message loss* per node from computing the ratio between the total missing messages and the total number of messages received successfully.

#### 2.1.1.3. Experimental Results

The results of the SYNC algorithm are given in Table 2.1. When increasing the number of nodes from 4 to 8, the total throughput drops by roughly 1.97 kbps per node (71.3kbps instead of 79.2kbps). There are two main reasons of the low throughput: firstly, each node needs to reserve the time for the refractory period (60 ms) within each period and, secondly, the *slotTime_EndNode* has to be set to at least 10 ms per slot time, per node.

| Nodes | 4 | 8 | 12 | 16 |
|---|---|---|---|---|
| Total Throughput (kbps) | 79.2 | 71.3 | 65.3 | 39.3 |
| Normalized, % | 93.4 | 84.1 | 76.5 | 46.4 |
| Max Individual (kbps) | 19.8 | 9.0 | 5.4 | 6.3 |
| Min Individual (kbps) | 19.7 | 8.7 | 5.4 | 1.4 |
| Message Loss (%) | 0.0 | 0.0 | 0.0 | 4.9 |

Table 2.1: SYNC's performance for different number of nodes, the maximum data rate at one single node was 84.8kbps.

Table 2.1 shows that the network performance of the SYNC algorithm becomes dramatically worse when the number of nodes is more than 12. We found the message loss is high in the case of 16 nodes and there is certain imbalance between the maximum and minimum individual node throughput. This is due to collisions of fire messages when increasing the number of nodes, despite the usage of the RFA mechanism.

## 2.2 Desynchronization

This section describes the design and implementation of a decentralized self-organizing *desynchronization* scheme to obtain distributed TDMA behaviour via the IEEE802.15.4 MAC. In addition, experimental results are presented for this algorithm using TinyOS iMote2 wireless sensor nodes. We first examine the implementation of the single-hop DESYNC algorithm [8] and then attempt to expand

this DESYNC algorithm to the multi-hop network case, by considering the two-hop neighbours (Section 4.1).

In the single-hop case, each node will broadcast a fire message every period and all other nodes can hear this message. Since each node receives all broadcast fire messages, all nodes in the topology will adjust their own fire message broadcast schedule to be performed at the middle of the two fire message times immediately preceding and following their own message broadcast. This means that, in the DESYNC algorithm, nodes in the network try to adapt their phase to be at the midpoint of the previous and next neighbor's phase. Once the WSN converges to the steady state, each TDMA slot time will occur between each node's own fire time and their next-phase neighbor fire time. In other words, each TDMA slot is equal to the time between a node's previous-phase neighbor and the node's own phase.

### 2.2.1 DESYNC-TDMA Algorithm and Implementation Description

Similarly to the case of the SYNC algorithm, our description is separated into two parts, before and after convergence to TDMA is achieved. Before convergence to TDMA is achieved, each node adjusts its phase and also updates its fire time. After convergence is obtained, every node in the network transmits fire messages periodically, i.e. in the steady state. During this stage, nodes can transmit data messages.

#### 2.2.1.1 Operation of DESYNC-TDMA during the convergence period

In DESYNC, each node $n_i$ $(1 \leq i \leq W)$ picks a particular time instant $t_i$ in which to broadcast its fire message based on the broadcasts of $n_{i-1}$ and $n_{i+1}$ (where we set $n_{-1} \equiv n_W$, $t_{-1} \equiv t_W$ and $n_{W+1} \equiv n_1$, $t_{W+1} \equiv t_1$). The determination of this time instant is performed immediately after the node detects the fire message of $n_{i+1}$, as shown in Figure 2.13. Hence, $n_i$ listens to all other nodes' fire message broadcasts and for the $k$th iteration (period) is set to fire according to the reactive listening primitive:

$$t_i^{(k)} = T + (1-\alpha)t_i^{(k-1)} + \alpha \frac{t_{i-1}^{(k-1)} + t_{i+1}^{(k-1)}}{2} \tag{2.1}$$

where $T$ is the desired TDMA period (in s) and $\alpha \in (0,1)$ a parameter that scales how far $n_i$ moves from its current fire time (at $t_i^{(k-1)}$) toward the desired midpoint. As mentioned at the beginning of this chapter, $t_i^{(k)}$and $t_i^{(k-1)}$ comprise the times when node $n_i$ transmitted its fire message within the $(k)$th and $(k-1)$th period (respectively), while $t_{i-1}^{(k-1)}$ and $t_{i+1}^{(k-1)}$ comprise the times when node $n_i$ received fire messages from nodes $n_{i-1}$ and $n_{i+1}$ (respectively) within the $(k-1)$th period.

Previous work [8][15] showed that the reactive listening primitive of (2.1) leads to near-optimal TDMA behavior in SS for 1-hop networks after $k_{\text{ss}}$ periods, which is expressed mathematically by:

$$\left| t_i^{(k_{\text{ss}})} - t_i^{(k_{\text{ss}}-1)} - T \right| < b_{\text{thres}} T \tag{2.2}$$

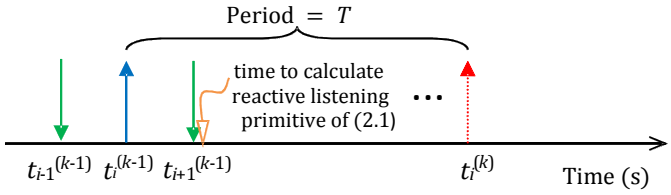


Figure 2.13: Scheduling of the fire-message broadcast for node $i$.

with $b_{\text{thres}}T$ ; $b_{\text{thres}} \epsilon$ [0.001,0.020] a preset threshold. In the steady-state, each node transmits data packets for $\frac{T}{W}$s immediately following its fire-message broadcast. If a node joins or leaves the network, thereby leading to $W' \neq W$ fire-message broadcasts, the remaining nodes reconfigure their fire-message broadcasts to converge to a new TDMA state and then continue data transmission once (2.2) is satisfied. Once TDMA behavior is achieved, the only overhead stems from the fire-message broadcasts, which include the number of the broadcasting node. Assuming negligible propagation delay and error-free transmission of broadcasts, $k_{\text{ss}}$ can be found by iterating the system's linear mapping matrix [8] until (2.2) is satisfied. We provide further details on this issue in our contribution in the next chapter of this thesis.

The format of the fire message consists of the node ID and sequence number of the fire message as shown in Figure 2.14.

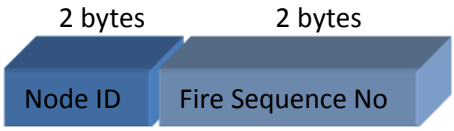


Figure 2.14: Format of a fire message for DESYNC algorithm.

Concerning implementation, as shown in Figure C.4 in Appendix C, the process begins with the node broadcasting its fire message every period via the event *AlarmFired*. When the node receives a fire message (*FireMessage.Received*), it will classify that fire time to be the next node ($t_{i+1}^{(k-1)}$) or the last node before its own firing ($t_{i-1}^{(k-1)}$) based on the *just_Fired* status. If it is in the case of the next node's fire time, the task *CalculatedNextFireTime* will be called and then the node will compute its own next fire time.

### 2.2.1.2 Operation of DESYNC-TDMA after the convergence is achieved

Each node will start sending data after checking for convergence. While a node receives $t_{i+1}^{(k-1)}$, it will check the time difference between $t_i^{(k)}$ and $t_i^{(k-1)}$ as shown in Figure 2.15. If this difference is less than $b_{\text{thres}}$, the node will assume that DESYNC has converged. After this point, the node will transmit data messages to a receiver node immediately after broadcasting its fire message. Those data messages are from the data in the flash memory of the TinyOS-based mote. When the node sends a data packet, it checks the remaining time of its TDMA slot. If the remaining time is less than the *time_SendData_onePack*, the node will stop sending data. Besides transmitting and receiving data, each node has to write the received data in flash memory. While the node is receiving data messages, it will keep them in received buffer. Once this node acquires a fire message from another node, it will write the buffered data to the flash memory immediately.

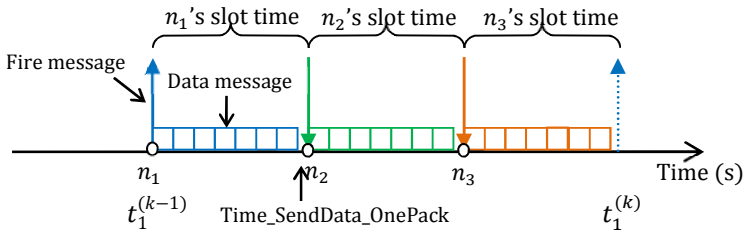


Figure 2.15: Local view of $n_1$ during DESYNC-TDMA and including data transmission.

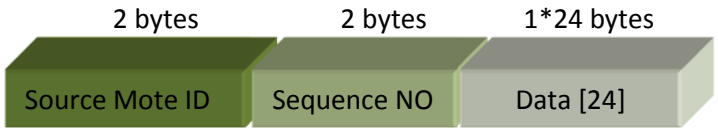

Figure 2.16: Format of a data message for DESYNC algorithm.

Concerning the format of the data message (Figure 2.16), we changed the size of the data length for iMote2 to 28 bytes. This data message includes the mote ID, the message sequence number and the data, which were read from the flash memory of the sender node.

We check the state of convergence at the time of the calculation of $t_i^{(k)}$. The check consists of examining the time difference between the node's $t_i^{(k)}$and $t_i^{(k-1)}$. This time difference is compared to the period ($T$ s). Under TDMA, the time difference between consecutive firings should be nearly equal to the period; we set the acceptable difference threshold between the two to be $0.01T$ in (2.2).

As shown in Figure C.5 in Appendix C, when the event *FireMessage.SendDone* of a fire message happens, the node will check the *set_converged* status. If the node is in the converged state, it will establish the receiver ID and then send data to that node. Similar to the SYNC case, the data is taken from the flash memory, starting at address 0x0030 0000. Transmitting will continue by checking the *time_SendData_onePack* within a few milliseconds after the event *DataMessage.SendDone* of a data message. If the current time approaches the expected *NextNode FireTime*, $(t_{i+1}^{(k-1)} + T)$, the node will stop sending data immediately.

Afterwards, when the event *DataMessage.Received* occurs, the node will start buffering the incoming data packets in the "received" buffer and, as mentioned, it will write the received buffer in the flash memory once the next fire message is received. We define the received buffer size to be 20 KB.

## 2.2.2 Experimental Validation

The DESYNC-TDMA algorithm was implemented on iMote2 wireless sensor motes following the hardware and settings used for the SYNC algorithm.

### 2.2.2.1 Experimental Setup

We tested with a single-hop network consisting of (maximally) 16 nodes. One mote was set up as a base station, recording all messages (fire messages and data messages) within a 1-minute (60 s) interval. We used the same fixed parameters which are:

- period ($T$) set to 2 second and
- alpha ($\alpha$) set to 0.85.

The number of nodes was set at 4, 8, 12 and 16. Five tests of 60 s were performed for each total number of nodes. The maximum throughput was found to be almost identical to the SYNC case, i.e. 85.3 kbps. We consider as the *total throughput* the measured data messages from each node (which are picked up by the base station) and as *normalized* (%) *throughput* the ratio between the total throughput and the maximum throughput. The *max* and *min individual throughput* was calculated from the average of the best and worst throughput per node for all 5 runs. Since all data messages include the source node ID and sequence number, we can evaluate the *message loss* by computing the ratio between the total missing messages and the total number of messages received correctly.

#### 2.2.2.2 Experimental Results

| Nodes | 4 | 8 | 12 | 16 |
|---|---|---|---|---|
| Total Throughput (kbps) | 84.0 | 80.5 | 78.5 | 75.2 |
| Normalized, % | 98.5 | 94.3 | 92.0 | 88.1 |
| Max Individual (kbps) | 21.1 | 10.1 | 6.6 | 4.8 |
| Min Individual (kbps) | 20.9 | 9.9 | 6.4 | 4.4 |
| Message Loss (%) | 0.0 | 0.0 | 0.0 | 0.0 |

Table 2.2: DESYNC -TDMA's performance for different number of nodes; the max. data rate with single transmitter and receiver setup was 85.3 kbps.

Table 2.2 shows the DESYNC-TDMA throughput and message loss. When the number of nodes increases, the total throughput measured in the network decreases by (approximately) 0.8 kbps per node. The table shows that we virtually had no message losses throughout our experiments. The balance between the maximum and minimum individual node throughput is quite good for all cases of

the number of nodes when comparing with the SYNC algorithm (Table 2.1), especially for the case of 16 nodes. This means DESYNC algorithm attains better performance for collision-free TDMA in WSNs than the SYNC algorithm.

## 2.3 PCO-based Inhibitory Coupling

This algorithm was proposed recently based on the original PCO model in [9]. It provides increased performance for desynchronization for distributed TDMA. The PCO scheme is able to provide for collision-free TDMA, where the fire times of all nodes are scheduled with spacing equal to $\frac{T}{W}$s.

When a single node exists in the network, it emits a pulse periodically every $T$ periods. Following the PCO principle, when the phase of node $i$ ($\varphi_i$) reaches 1, the node will emit a pulse and reset its phase back to 0. To achieve the PCO with inhibitory coupling, a concave function is introduced, which typically is [9]:

$$f(\varphi_i^{(k)}) = -\log(\varphi_i^{(k)}) \tag{2.3}$$

Inhibitory coupling defines the coupling constant as:

$$\varepsilon = -\log(1-\alpha) \tag{2.4}$$

where $\alpha \in (0,1)$ is the phase-coupling constant controlling the speed of the phase adaptation and $\varepsilon > 0$ is the coupling strength. The protocol achieves the strict desynchronization based on the following updating rule:

$$\varphi_i^{(k)} = \begin{cases} f^{-1}((f(\varphi_i^{(k-1)}) + \varepsilon) & \text{if } \varphi_i^{(k-1)} \in (1-\frac{1}{W}, 1) \\ \varphi_i^{(k-1)} & \text{otherwise} \end{cases} \tag{2.5}$$

where the factor $W \geq 1$ controls the spacing between nodes

$$f(\varphi_i^{(k)}) = -\ln(1 - W(1-\varphi_i^{(k)})) \tag{2.6}$$

Taking the exponential function for both sides, we have:

$$e^{f(\varphi_i^{(k)})} = e^{-\ln(1-W\left(1-\varphi_i^{(k)}\right))} \tag{2.7}$$

which leads to:

$$e^{-f(\varphi_i^{(k)})} = (1 - W(1-\varphi_i^{(k)})) \tag{2.8}$$

From (2.5), when the node will update its phase, we have:

$$f^{-1}((f(\varphi_i^{(k-1)}) + \varepsilon) = (1 - \frac{1}{W}(1 - e^{-f(\varphi_i^{(k-1)})}e^{-\varepsilon})) \tag{2.9}$$

From (2.4) we get: $e^{-\varepsilon} = 1 - \alpha$. Combining this with (2.8) leads to:

$$f^{-1}((f(\varphi_i^{(k-1)}) + \varepsilon) = (1 - \frac{1}{W}((1-\alpha)W(1-\varphi_i^{(k-1)}) + \alpha) \quad (2.10)$$

Therefore:

$$\varphi_i^{(k)} = (1-\alpha)\varphi_i^{(k-1)} + \alpha(1 - \frac{1}{W}) \quad (2.11)$$

The last derivation describes the practical phase update scheme for PCO-based desynchronization with inhibitory coupling. As noted in [9] and shown in (2.11), this phase update depends on the number of nodes ($W$) in the WSN.

### 2.3.1 Implementation Details

PCO-based desynchronization with inhibitory coupling adjusts the period of fire message broadcasts of each node according to the received fire messages within a certain interval in-between each node's own consecutive firings [9]. As such: *(i)* the scheduled time for the fire message broadcast of each node changes after *each* message received within a certain time interval [i.e. adaptation is not based on the previous and next fire message as in (2.1)]; *(ii)* knowledge of the total number of nodes ($W$) is required [9].

We implemented the PCO-based inhibitory coupling of the previous subsection in iMote2 wireless sensors. The basic time diagram of the operation is shown in Figure 2.17. In this section we explain the details of the implementation of the scheme before the convergence to TDMA. For the operation *after* convergence to distributed TDMA is achieved, the process follows the DESYNC algorithm of Subsection 2.2.1.2.

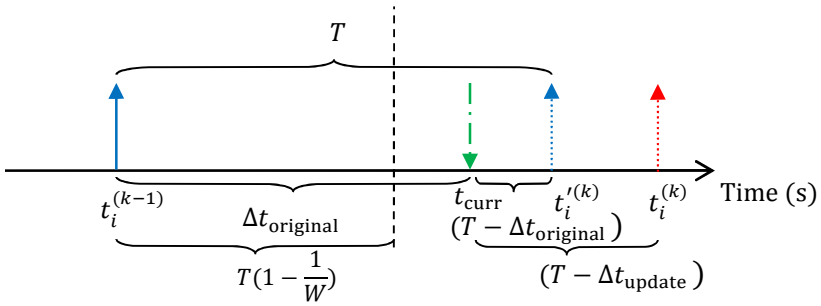


Figure 2.17: Local view of PCO-based Inhibitory Coupling.

Typically, each node will fire every period ($T$). Once a node receives a fire message from another node within its "listening" interval $[(T - \frac{1}{W}T, T)]$, it will

update its next fire time with (2.12), which is directly derived from the theoretical (2.11):

$$\Delta t_{\text{update}} = (1-\alpha)\,(t_{\text{curr}} - t_i^{(k-1)}) + \alpha \times T(1-\frac{1}{W}) \tag{2.12}$$

Once a node knows the total number of nodes, it can precalculate the second term of the summation involved in this equation to facilitate the implementation. The node's fire message time will be updated based on $\Delta t_{\text{update}}$ calculated for all received messages from all nodes *i* as shown in (2.13). This update is applied only if $t_{\text{curr}} - t_i^{(k-1)} > T(1-\frac{1}{W})$ and $t_{\text{curr}} - t_i^{(k-1)} < T$

$$t_i^{(k)} = t_{\text{curr},i} + (T - \Delta t_{\text{update},i}) \tag{2.13}$$

All fire messages received outside the "listening interval" of $(T-\frac{1}{W}T, T)$s are simply ignored. After $k_{\text{ss}}$ phase updates, such an approach has been shown [9] to converge to "dispersed" fire message broadcasts at intervals of $\frac{1}{W}T$s within consecutive firings of each node.

We define the short fire message format similar to the DESYNC algorithm because of collision-free in communication. The format of this message consists of the node ID and sequence number of the fire message as shown in Figure 2.18.

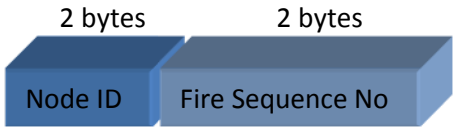


Figure 2.18: Format of a fire message for PCO-based inhibitory coupling.

In the beginning, the node fires its broadcast message every period based on the event *Time.AlarmFired*. Each node is interested in the fire message received within the "listening" phase. In fact, if the received fire message is received outside of the listening phase, the node will not update its fire message scheduling. On the other hand, if the node receives a fire message within the listen phase, it will call task *setAlarmFired* to shift the new next fire time as illustrated in the Figure 2.19.

### 2.3.2 Experimental Validation

The PCO-based inhibitory coupling algorithm (PCO-DESYNC) was implemented on iMote2 wireless sensor motes following the experimental setup used for SYNC and DESYNC.

We used the same (fixed) parameters as before, which are

- period ($T$) set to 2s and
- alpha ($\alpha$) set to 0.85

for all measurements. The number of nodes was selected to be 4, 8, 12 and 16 and five tests of 60s were executed per case. The maximum throughput in PCO-based inhibitory coupling algorithm is similar to the DESYNC algorithm.

| Nodes | 4 | 8 | 12 | 16 |
|---|---|---|---|---|
| Total Throughput (kbps) | 82.1 | 75.8 | 70.3 | 64.2 |
| Normalized, % | 96.3 | 88.8 | 82.4 | 75.2 |
| Max Individual (kbps) | 20.8 | 9.6 | 6.0 | 4.1 |
| Min Individual (kbps) | 20.1 | 9.3 | 5.7 | 3.9 |
| Message Loss (%) | 0.0 | 0.0 | 0.0 | 0.0 |

Table 2.3: Performance of PCO-based inhibitory coupling for different number of nodes; the maximum data rate at the single transmitter-receiver setup was 85.3 kbps.

The PCO-DESYNC throughput and message loss shows in Table 2.3. When the number of nodes increases, the total throughput decreased by approximately 1.5kbps per node. This throughput's decrease was almost twice as the DESYNC throughput's reduction for the corresponding cases (Table 2.2). That makes the PCO-DESYNC bandwidth efficiency to be lower than that of DESYNC for all cases of the number of nodes. This is due to the more frequent updating of each node's fire time based on the listening interval, which makes PCO-DESYNC more prone to noise in the fire message times than the DESYNC algorithm. However, the table also shows the message loss was virtually zero and that PCO-DESYNC remains significantly superior to the SYNC algorithm (Table 2.1).

## 2.4 Conclusion

In order to design an efficient distributed synchronization protocol, the achieved data throughput must be maximized and collisions during the steady state of the operation must be minimized. We have presented, implemented and experimentally validated the performance of three different protocols for distributed TDMA at the IEEE802.15.4 MAC. All protocols use broadcastsynchronization (fire) messages from each node, based on each node's own local clock. The first algorithm, SYNC, was found to have the following detriments:

(i) Setting up the reachback response and the slot time of each node in order to avoid collisions during the TDMA (steady) state is cumbersome to implement in the standard IEEE802.15.4 MAC.

(ii) If the refractory period is kept short, the protocol becomes more efficient in the steady state but synchronization becomes difficult; alternatively, if the refractory period grows, the protocol becomes less efficient.

(iii) The total throughput of SYNC approach drops dramatically when the number of nodes increases as shown in Figure 2.19.

For DESYNC and PCO-DESYNC, which were found to be the most efficient cases (both communication-wise but also implementation-wise), we will be able to identify the trends of the convergence time for the distributed synchronization in the next chapter. Therefore an interesting problem would be to derive stochastic estimates of the convergence time for these two cases of desynchronization. Specifically, we are interested in establishing the *settling time* to achieve TDMA convergence in WSNs based on DESYNC and PCO-DESYNC. This is the topic of the following chapter.

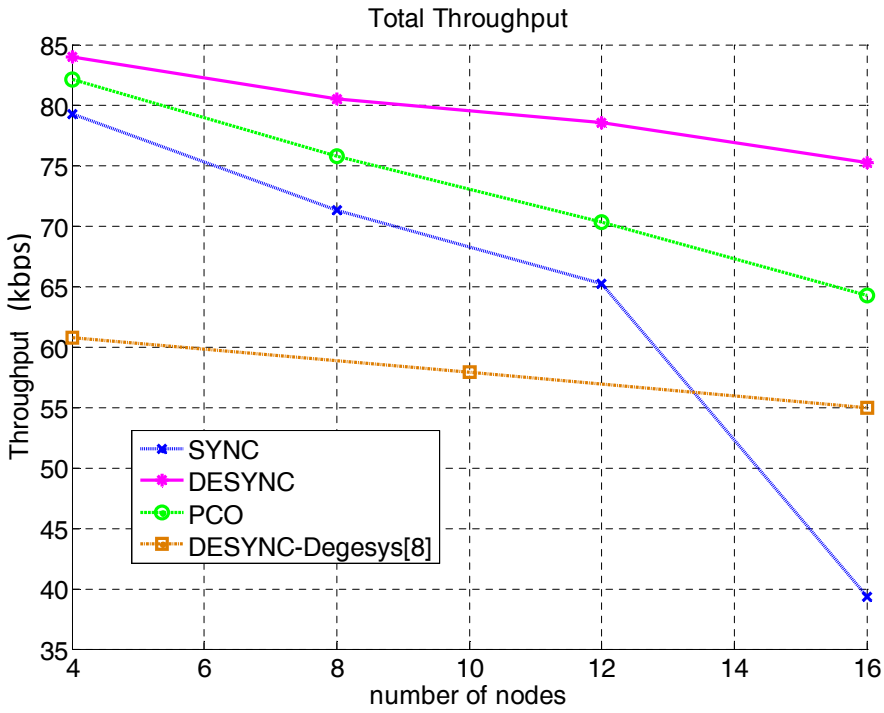


Figure 2.19: Results of the total throughput for the different protocol of the synchronization primitive.

# Chapter 3
# Stochastic Modeling of Convergence to Desynchronization in Wireless Sensor Networks

Following the description of the previous chapter, desynchronization can be abstracted as the algorithm for revising the fire message broadcast time of each wireless sensor node (or "node") of a fully-connected WSN based on the received messages from the other nodes within a certain time period. As introduced in Chapter 2, we consider a network of $W$ fully-connected WSNs, each with an internal clock with $T$s period and follow the notational conventions of that chapter with minor extensions[1]. Moreover, this model could also be applicable for the multi-hop WSNs as shown in Chapter 4. Each node keeps the *phase* of fire messages in relation to its clock ticks and, in the steady state, it broadcasts one fire message every $T$s as in Chapter 2 and without loss of generalization. The actual broadcast (or "fire") times of every node are determined based on the reactive listening primitives of the next two subsections. For expositional purposes, in the next section we ignore the phase measurement noise and assume each fire time can be determined precisely and instantaneously by all nodes. This noise is however taken into account in the modeling framework of the Section 3.3.

One of the major open problems of desynchronization in WSNs is the derivation of robust estimates for the required iterations until convergence to steady state is achieved. While previous work has derived order-of-convergence estimates, these are derived based on heuristics and are only expected to capture the asymptotic

[1] Italicized letters indicate scalars and boldface letters indicate vectors. For vectors a and b, the circular convolution [94] with period $W$ is given by $(0 \le n < W)$: $(\mathbf{a} * \mathbf{b})_W[n]$. Random variables (RVs) are represented by Greek uppercase letters, e.g. $\Phi \sim N(\mu_\Phi, \sigma_\Phi)$ or $\Delta \sim U(\mu_\Delta, \sigma_\Delta)$, with $N()$ and $U()$ reserved to indicate the normal and uniform PDFs, respectively, with mean $\mu_\Phi$ (or $\mu_\Delta$) and standard deviation $\sigma_\Phi$ (or $\sigma_\Delta$).

behavior of the desynchronization process. Instead, in this chapter we propose to estimate the convergence iterations of desynchronization by embracing the non-deterministic aspects of this process and utilize stochastic (instead of deterministic) estimates for the convergence iterations of desynchronization. That is, we do not use the actual firing times of the WSN nodes, but rather their statistical description, which does not require any explicit assumption on the node firing order. In order for our estimates to have wide applicability, we focus on the two reactive listening primitives that form the basis of all desynchronization algorithms and have been described in detail in the previous chapter: *(i)* the DESYNC algorithm of Degesys, Patel *et al* [8][13]; *(ii)* PCOs with inhibitory coupling, as proposed by Pagliari *et al* [9], which, under the knowledge of the total number of nodes, have been conjectured to converge to steady state faster than the DESYNC algorithm [9]. Via the proposed stochastic estimation framework, we derive analytic formulations for the number of iterations until the firing (or pulsing) time is *expected to have converged* to within $b_{\text{thres}}T$s from its SS value, with $b_{\text{thres}} \in [0.001,0.020]$, i.e. guard times of 1-20ms for firing cycles with period $T = 1$s. In addition, we validate our results based on a real WSN deployment as well as under a simulation environment and demonstrate the accuracy of the proposed stochastic estimates against the convergence bounds found in the literature [101][118].

For measurement and analysis of convergence properties, a base station can be used to passively listen to all fire message broadcasts, with $W$ consecutive firings comprising a *firing cycle*. For all desynchronization algorithms, it is immaterial which physical sensor node is linked to which firing, as desynchronization is solely dependent on the received fire message times [8][9][13][15][34][81]-[83][107]. For this reason, we shall be explicitly discussing firing events and not the physical nodes that create them.

As shown in Figure 3.1, we can imagine the nodes as beads moving clockwise on a ring [8] with period $T$=1s. When node $n_i$ reaches the top, it fires: its phase has

reached one and it is reset to zero once the fire message has been broadcast. During each firing cycle, each firing node $n_i$ increments its phase and the phase of received fire messages by

$$\forall i,k: \varphi_i^{(k)} \leftarrow \varphi_i^{(k)} + \Delta t \pmod 1. \tag{3.1}$$

with $\Delta t$ depending on the node's internal clock granularity. Via the received fire messages, each node $n_i$ adjusts its firing phase $\varphi_i^{(k-1)}$ to $\varphi_i^{(k)}$ based on the reactive listening primitives of the next two subsections. Thus, $k$ indicates the $k$th *phase-update iteration* and not the $k$th firing cycle. The two *may* or *may not* coincide for each desynchronization algorithm. In this section we ignore the phase measurement noise and assume each fire time can be determined precisely by all nodes. This noise is however taken into account in the modeling framework

| Symbol | Definition |
|---|---|
| $W$ | total number of nodes in the desynchronization process |
| $T$ | period of firing cycles (in seconds) |
| $\alpha$ | phase coupling constant of desynchronization |
| $b_{\text{thres}}$ | steady-state convergence threshold of desynchronization |
| subscripts: $i, i-1, i+1$ | indicating that the variable corresponds to the current, previous or next firing |
| $\|\mathbf{v}\|$ | vector norm-2 |
| $\mathbf{v}[n]$ | the $n$th element of vector $\mathbf{v}$, $n \geq 0$ |
| $(\mathbf{a} * \mathbf{b})_W[n]$ | the $n$th sample of circular convolution of period $W$ |
| $\varphi^{(k)}$ | quantity $\varphi$ computed after $k$ iterations |
| expr (mod 1) | modulo-1 of expression expr $\in \mathbb{R}$ |
| $\lfloor u \rfloor$ | the largest integer that is smaller or equal to $u$ |
| $\lceil u \rceil$ | the smallest integer that is largest or equal to $u$ |
| Pr[expr] | probability of occurrence of expression expr |

Table 3.1: Nomenclature table

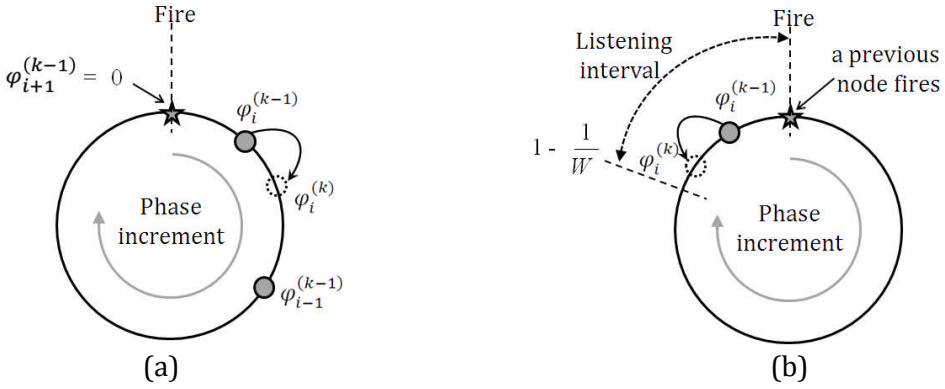


Figure 3.1: The $k$th phase update of node $n_i$ happens when: (a) node $n_{i+1}$ (next firing node) fires in DESYNC; (b) another node fires in PCO-based desynchronization and $n_i$ is within the listening interval (i.e. if $1 - \frac{1}{W} < \varphi_i^{(k-1)} < 1$).

## 3.1 Phase Domain of DESYNC

It is mathematically convenient to express the reactive listening primitive of (2.1) using the phase $\varphi_i^{(k)}$ in relation to the $k$th periodic interval [8] (Figure 3.2):

$$\varphi_i^{(k)} = \left[(1-\alpha)\varphi_i^{(k-1)} + \alpha \frac{\varphi_{i-1}^{(k-1)} + \varphi_{i+1}^{(k-1)}}{2} \pmod 1\right]. \tag{3.2}$$

In this approach, the $i$th firing node, $n_i$, updates its phase once within each firing cycle at the moment the next node ($n_{i+1}$) fires. As shown in Figure 3.1(a), the update uses the previous and next node's fire-message broadcast phase and moves $n_i$'s phase towards the middle of the interval between the firing of the previous and the next node by ($1 \leq i \leq W$). With $\alpha \in (0,1)$ the phase-coupling constant controlling the speed of the phase adaptation. Previous work [8][13] showed that the reactive listening primitive of (3.2) disperses all fire message broadcasts at intervals of $\frac{T}{W}$ within each periodic firing cycle. Thus, it leads to near-optimal TDMA in SS after $k_{\text{ss}}$ iterations of (3.2), where all fire messages are periodic and the phase update of (3.2) leads to convergence, expressed by:

$$\text{At } k_{\text{ss}}\text{th phase update: } \forall i: \left|\varphi_i^{(k_{\text{ss}})} - \varphi_i^{(k_{\text{ss}}-1)}\right| \leq b_{\text{thres}}, \tag{3.3}$$

with $b_{\text{thres}}$ a preset threshold, typically $b_{\text{thres}} \in [0.001, 0.02]$.

In SS, each node transmits data packets for $T(\frac{1}{W} - b_{\text{thres}})$s immediately following its fire-message broadcast (which limits the maximum number of nodes supported under collision-free TDMA to less than $\left\lfloor \frac{1}{b_{\text{thres}}} \right\rfloor$). If a node joins or leaves the network, the remaining nodes reconfigure their fire messages to converge to a new

TDMA state and then proceed with data transmission once (3.3) is satisfied. Once TDMA behavior is achieved, the only overhead stems from the fire-message broadcasts, which are very short packets (just two bytes in our implementation). Assuming negligible propagation delay and error-free detection of messages, it has been conjectured [8][13] that convergence to TDMA requires iterations of order:

$$k_{\text{DESYNC}[8][9]} \sim O\left(\frac{1}{\alpha} W^2 \ln\left(\frac{1}{b_{\text{thres}}}\right)\right). \tag{3.4}$$

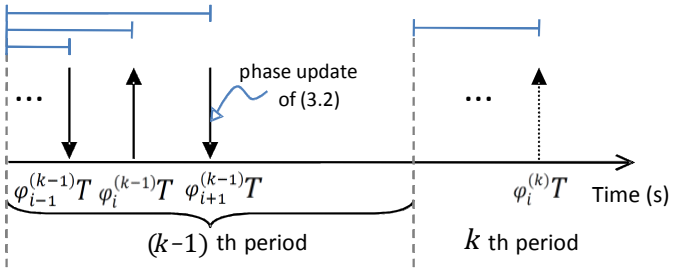


Figure 3.2: Scheduling of the $k$th fire-message broadcast for node $i$ for DESYNC.

## 3.2 PCO-based Inhibitory Coupling in Phase Domain

In PCO-based desynchronization, the phase of each node $n_i$, $\varphi_i^{(k-1)} \in [0,1)$, refers to the delay between $n_i$'s last firing and the reception time of the $k$th fire message; see Figure 3.3 for a pictorial example. PCO-based desynchronization with inhibitory coupling adjusts each node's phase according to the received fire messages within a certain interval in-between the node's own consecutive firings [9]. This is indicated as the "listening interval" in Figure 3.1(b). As such: *(i)* the phase of each node changes after each message received within the listening interval [i.e. a varying number of phase updates may occur within each firing cycle]; *(ii)* knowledge of the total number of nodes ($W$) is required [9]. Hence, for any period of any node $n_i$, when the $k$th fire message is received at $\varphi_i^{(k-1)}T$s after $n_i$'s last firing, with $1 - \frac{1}{W} < \varphi_i^{(k-1)} < 1$, the node's phase updating rule is [9]:

$$\varphi_i^{(k)} = [(1-\alpha)\varphi_i^{(k-1)} + \alpha(1 - \tfrac{1}{W}) \pmod 1] \tag{3.5}$$

which is directly derived from (2.12) with $\alpha \in (0,1)$ the phase-coupling constant controlling the speed of the phase adaptation. All fire messages received outside the "listening interval" $(1 - \frac{1}{W}, 1)$ are simply ignored. The update of (3.5) changes

the next fire time of $n_i$ from $T$ to $T(1+\varphi_i^{(k-1)}-\varphi_i^{(k)})$, as seen in Figure 3.3. After $k_{\text{ss}}$ phase updates, $\varphi_i^{(k_{\text{ss}})}=1-\frac{1}{W}\pm b_{\text{thres}}$ or, equivalently, (3.3) holds under convergence. Hence, like the DESYNC case, once such TDMA behavior is achieved, the only overhead stems from the short fire message broadcasts. Unlike DESYNC though, the phase adaptation in (3.5) requires the knowledge of the total number of WSNs, $W$. Assuming negligible propagation delay and error-free detection of broadcast messages and $1-\alpha>\frac{1}{W}$., it has been shown [9] that the number of firing cycles for convergence to TDMA requires is lower bounded by:

$$k_{\text{PCO[9]}} \geq \left\lceil \frac{\ln[\frac{b_{\text{thres}}}{2[1+\alpha^{-W}(1-\alpha)^{-1}]}]}{\ln(1-\alpha)+\ln(W)} \right\rceil, \tag{3.6}$$

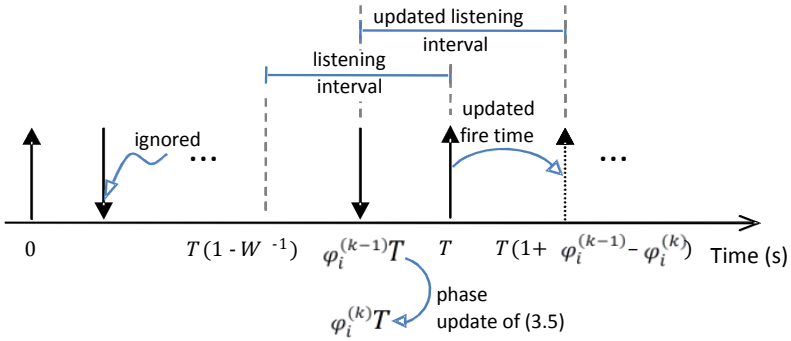


Figure 3.3: Phase adaptation during the reception of the ***k***th message in PCO-based desynchronization.

## 3.3 Stochastic Modeling of DESYNC and PCO-based Inhibitory Coupling

Our stochastic estimates of the convergence time for DESYNC-based and PCO-based desynchronization assume that the phase of transmitted or received fire messages is contaminated by white noise due to varying propagation and processing delays of a practical WSN environment. In addition, convergence is determined based on the PDF of a node's phase variable. These conditions are formalized by the following propositions.

*Definition 1* (*Convergence to Steady State*): The notion of phase update convergence to SS is determined based on the probability density function (PDF) of each node's current firing phase under successive DESYNC and PCO phase updates: the phase of the current firing evolves over time following (3.1) but its updates via

(3.2) or (3.5) converge according to (3.3). □

*Definition 2* (*Phase Model*): Each node's initial phase variables, $(\varphi_i^{(0)}, \varphi_{i-1}^{(0)}, \varphi_{i+1}^{(0)})$ when performing the phase update of (3.2) or (3.5) are modeled as independent random variables that are uniformly distributed between $[0, 1)$, i.e. $\forall \text{fire} \in \{i, i-1, i+1\}$: $\Phi_{\text{fire}}^{(0)} \sim \text{P}_{\Phi_{\text{fire}}^{(0)}}$ with

$$\text{P}_{\Phi_{\text{fire}}^{(0)}} = U(\mu_{\Phi_{\text{fire}}^{(0)}}, \sigma_{\Phi^{(0)}}) \pmod 1.$$

Phase variables refer to nodes' *own fire time* for DESYNC and to nodes' *received fire times* for PCO-based desynchronization. We define the mean times of successive phase updates to be equidistant, which, for DESYNC, is expressed as:

$$\forall i < W: \mu_{\Phi_{i-1}^{(0)}} - \mu_{\Phi_i^{(0)}} = \frac{1}{W},\ \mu_{\Phi_i^{(0)}} - \mu_{\Phi_{i+1}^{(0)}} = \frac{1}{W} \tag{3.7}$$

with $\mu_{\Phi_0^{(0)}} = \mu_{\Phi_W^{(0)}}$ and for the PCO update of (3.5) is stated by:

$$\forall i: \mu_{\Phi_i^{(0)}} = 1 - \frac{1}{W} \tag{3.8}$$

In the beginning of the desynchronization, the nodes are assumed as completely uncoordinated, i.e. $\forall \text{i}: \sigma_{\Phi^{(0)}} = \frac{1}{\sqrt{12}}$. □

Our estimates of the convergence iterations for DESYNC and PCO-based desynchronization assume each phase in (3.2) and (3.5) is contaminated by white noise due to the varying propagation and processing delays of a WSN environment.

*Definition 3* (*Measurement Noise Model*): All phase values in the update of (3.2) or (3.5) are contaminated by additive noise, modeled as an independent, zero-mean, uniformly-distributed, random variable, $\Delta \sim U(0, \sigma_\Delta)$. □

The standard deviation of the measurement noise of Definition 3 will be derived experimentally, as this includes the effects of propagation and processing delays that can only be inferred via measurements from a real deployment.

Due to the measurement noise and the interaction between nodes, for each phase update $k$, the PDF of the phase of any node $n_i$, $\text{P}_{\Phi_i^{(k)}}$, changes after applying (3.2) or (3.5); consequently, this changes the probability of convergence to SS:

$$\Pr\left[\left|\Phi_i^{(k)} - \mu_{\Phi_i^{(k)}}\right| \leq b_{\text{thres}}\right] = \int_{-b_{\text{thres}}}^{b_{\text{thres}}} \text{P}_{\Phi_i^{(k)}}(u - \mu_{\Phi_i^{(k)}}) du$$

$$= \operatorname{erf}\left(\frac{b_{\text{thres}}}{\sqrt{2}\sigma_{\Phi_i^{(k)}}}\right) \tag{3.9}$$

with $\operatorname{erf}(u)$ the error function [14]. Notice that (3.9) holds under the assumption that $\mathrm{P}_{\Phi_i^{(k)}}$ converges to a normal distribution for both DESYNC and PCO-based desynchronization, which, as the next two subsections will show, turns out to be the case. Consequently, we use a stochastic criterion for convergence based on the confidence intervals of the normal distribution [14]. By defining the confidence coefficient

$$c_{\text{conf}} = \Pr\left[\left|\Phi_i^{(k)} - \mu_{\Phi_i^{(k)}}\right| \leq b_{\text{thres}}\right], 0 < c_{\text{conf}} < 1, \tag{3.10}$$

we have from (3.9):

$$\sqrt{2}\operatorname{erf}^{-1}(c_{\text{conf}})\,\sigma_{\Phi_i^{(k)}} - b_{\text{thres}} = 0 \tag{3.11}$$

with $\operatorname{erf}^{-1}(u)$ the inverse error function that can be expressed by its Maclaurin series:

$$\operatorname{erf}^{-1}(u) = \tfrac{1}{2}\sqrt{\pi}\left(u + \tfrac{\pi}{12}u^3 + \tfrac{7\pi^2}{480}u^5 + \tfrac{127\pi^3}{40320}u^7 + \cdots\right). \tag{3.12}$$

Thus, (3.11) becomes the mechanism for defining the phase update iteration leading to SS: we determine the phase-update iteration $k_{\text{ss}}$ for which the amplitude of the left side of (3.11) is minimized, i.e. the phase update iteration leading to convergence with probability that closely matches $c_{\text{conf}}$, which is our (predetermined) confidence.

*Definition* 4 (*Converged State with* $c_{\text{conf}} \times 100\%$ *Confidence*, $0 < c_{\text{conf}} < 1$): We define a desynchronization mechanism as being in "steady state" or "converged state" with $c_{\text{conf}} \times 100\%$ confidence, at the $k_{\text{ss}}$th phase-update iteration, where:

$$\forall i\colon\ k_{\text{ss}} = \arg\min_{\forall k \in \mathbb{N}} \left|\sqrt{2}\operatorname{erf}^{-1}(c_{\text{conf}})\,\sigma_{\Phi_i^{(k)}} - b_{\text{thres}}\right| \tag{3.13}$$

with $\sigma_{\Phi_i^{(k)}}$ the standard deviation of the phase PDF of the current firing at the $k$th iteration of (3.2) or (3.5) and $\operatorname{erf}^{-1}(u)$ given by (3.12). □

Since $\sigma_{\Phi_i^{(k)}}$ is affected by the measurement noise, in order for the system to remain in the converged state indefinitely, the threshold for the convergence, $b_{\text{thres}}$, must be set according to the (estimated) $\sigma_\Delta$. Conversely, we can treat the

entire desynchronization process as a "black box" system and estimate $\sigma_\Delta$ by measuring the phase deviation from the mean obtained when performing the update of (3.2) or (3.5) during SS. This will be demonstrated in the experimental section.

### 3.3.1 Modeling of Firing Cycles Required for DESYNC's Convergence Time

**Proposition 1**: *Under the setup of Definition 2 and Definition 3, the number of firing cycles for the DESYNC of (3.2) to converge under Definition 4 is*:

$$k_{\text{desync}} = \arg\min_{\forall k \in \mathbb{N}} \left| \sqrt{2} \times \text{erf}^{-1}(c_{\text{conf}})\, \sigma_{\text{desync}}^{(k)} - b_{\text{thres}} \right| \tag{3.14}$$

with

$$\sigma_{\text{desync}}^{(k)} = \sqrt{\|\mathbf{v}^{(k)}\|^2 \sigma_{\Phi^{(0)}}^2 + \sum_{j=1}^{k} \|\mathbf{v}^{(j)}\|^2 \sigma_\Delta^2}, \tag{3.15}$$

$$\mathbf{v} = \left[\frac{\alpha}{2} \quad 1-\alpha \quad \frac{\alpha}{2}\right] \text{ and } \mathbf{v}^{(j)} = \underbrace{\mathbf{v} * \ldots * \mathbf{v}}_{j \text{ times}} \tag{3.16}$$

the vector produced by $j$ consecutive circular convolutions of length $W$ (by zero-padding $v$ to length $W$).

*Proof*: By denoting all input random variables by $\mathbf{\Phi}^{(0)} = \left[\Phi_1^{(0)} \quad \ldots \quad \Phi_W^{(0)}\right]$ and the corresponding additive measurement noise sources [independent identically distributed (iid) random variables] from Definition 3 by $\mathbf{\Delta} = [\Delta_1 \quad \ldots \quad \Delta_W]$, the first iteration of the phase update process of (3.2) is:

$$\mathbf{\Phi}^{(1)} = \left[\mathbf{v} * \left(\mathbf{\Phi}^{(0)} + \mathbf{\Delta}\right) \pmod 1\right]. \tag{3.17}$$

Thus, we have: $\forall i: \mu_{\Phi_i^{(1)}} = \mu_{\Phi_i^{(0)}}$ and

$$\forall i: \sigma_{\Phi^{(1)}} = \|\mathbf{v}\| \sqrt{(\sigma_{\Phi^{(0)}}^2 + \sigma_\Delta^2)}. \tag{3.18}$$

Generalizing this for $k$ iterations, we reach: $\mu_{\Phi_i^{(k)}} = \mu_{\Phi_i^{(0)}}$ and $\sigma_{\Phi^{(k)}} = \sigma_{\text{desync}}^{(k)}$, shown in (3.9). We can now make the following observations:

- Each term $\Phi_i^{(k)}$ is a linear mixture of independent random variables (i.e. iid noise $\mathbf{\Delta}$ and phase vector $\mathbf{\Phi}^{(0)}$);
- $\forall i \in \mathbb{N}^*$, we can pick $\varepsilon = (1-\alpha)^k \sigma_{\Phi^{(0)}}$ and, from (3.15), $\sigma_{\text{desync}}^{(k)} > \varepsilon$;
- All initial PDFs have finite support (they are all variants of the uniform distribution); hence, densities $\text{P}_{\Phi_i^{(k)}}$ will have finite support since they are

linear mixtures of PDFs with finite support.

These three observations satisfy the three conditions for the generalized form of the central limit theorem (CLT) to be applicable [[14], pp. 219-220] (see also Appendix 1), and thus:

$$\Phi_i^{(k)} \sim N(\mu_{\Phi_i^{(0)}}, \sigma_{\text{desync}}^{(k)}) \pmod 1 \tag{3.19}$$

Hence, we reach (3.14) for convergence under Definition 4. □

Proposition 1 shows that under the given algorithm of (3.2) for DESYNC, $k_{\text{desync}}$ is affected by $\alpha$, as well as by the noise assumptions, expressed by $\sigma_{\Phi^{(0)}}$ and $\sigma_\Delta$ in Definition 2 and Definition 3, respectively. Interestingly, the total number of nodes does not appear to influence the convergence to the steady state. For the special cases of $W \in \{2,3,4\}$ nodes, we set $W = 5$ in the circular convolution of (3.16) to avoid erroneous overlapping within $\mathbf{v}^{(j)}$ due to the short length of the circular convolution.

### 3.3.2 Modeling of Firing Cycles Required for Convergence in PCO-based Desynchronization with Inhibitory Coupling

**Proposition 2**: *Under the setup of Definition 2 and Definition 3, the number of firing cycles for convergence under Definition 4 in the PCO-based inhibitory coupling of (3.5) is*:

$$k_{\text{PCO}} = \arg\min_{\forall k \geq 2} \left| \sum_{l=2}^{k} u_l + 1 - \frac{1}{W} - k_{\text{update}} \right| \tag{3.20}$$

With

$$u_l = \text{erf}\left( \frac{\left\lfloor \frac{W}{2} \right\rfloor + 1}{W \sigma_{\text{PCO}}^{(l-1)} \sqrt{2}} \right) - \text{erf}\left( \frac{1}{W \sigma_{\text{PCO}}^{(l-1)} \sqrt{2}} \right) \tag{3.21}$$

$$k_{\text{update}} = \arg\min_{\forall k \in \mathbb{N}} \left| \sqrt{2} \times \text{erf}^{-1}(c_{\text{conf}})\, \sigma_{\text{PCO}}^{(k)} - b_{\text{thres}} \right| \tag{3.22}$$

$$\sigma_{\text{PCO}}^{(k)} = \sqrt{(1-\alpha)^{2k} \sigma_{\Phi^{(0)}}^2 + \frac{(\alpha-1)^2}{\alpha(\alpha-2)} [(1-\alpha)^{2k} - 1] \sigma_\Delta^2}, \tag{3.23}$$

*Proof:* We separate the proof into a series of stages based on the temporal evolution of the convergence process.

**First firing cycle:** The expected number of phase updates within the first firing

cycle is equal to the number of firings expected to be heard within $(1-\frac{1}{W},1)$ from the initiation of each firing phase, which is:

$$\sum_{j=1}^{W-1} j\binom{W-1}{j}\left(\frac{1}{W}\right)^{j}\left(1-\frac{1}{W}\right)^{W-1-j}=1-\frac{1}{W} \tag{3.24}$$

*Concerning the first firing:* What is important in our modeling is not the change in the actual fire times but the change in the statistics (i.e. first two moments) of the phase random variables in the update of (3.5). In PCO, the phase update iteration of (3.5) only uses the moments of the node's own phase when (3.5) is applied; hence it is using the moments of the previous phase update iteration. To calculate $A_{\text{init}}$ (expected number of firings heard in the first iteration) one must assume the initial moments for all phase variables of all nodes as there is no prior information on the node's firing. That is, it could be that the node is the first to fire or the last to fire within one firing cycle – there is no way to know this and thus we assume the statistics of the previous iteration for all nodes. In the case of $A_{\text{init}}$, that would be the initial moments as presented in Definition 2 and Definition 3, thereby deriving $A_{\text{init}}=1-\frac{1}{W}$.

We are interested in the estimation of the expected number of firing cycles for convergence to TDMA under the predefined threshold. However, what we can count via the stochastic analysis is: how many times are the nodes expected to apply (3.5) within each firing cycle. Then, through that we can estimate how many firing cycles will it take (on average) until (3.5) leads to convergence under Definition 4. Thus, the expected number of firings each node will receive during the first firing cycle is what is needed to find out how many times (3.5) (phase update) will be applied in the first firing cycle. This is given by $A_{\text{init}}$.

This is derived based on the binomial theorem, since Definition 2 mandates that $\forall i\colon \Phi_i^{(0)}$ is uniformly distributed within [0,1). Since $A_{\text{init}}$ approaches unity for large $W$, assuming sufficiently large $W$, each node will update its phase variable once in the first firing cycle via (3.5), thereby deriving:

$$\forall i\colon \Phi_i^{(1)}=(1-\alpha)(\Phi_i^{(0)}+\Delta_i^{(0)})+\alpha(1-\tfrac{1}{W}) \quad (\text{mod } 1) \tag{3.25}$$

with $\Delta_i^{(0)}$ being random variables modeling the measurement noise of $\varphi_i^{(0)}$ from

Definition 3 From (3.8) and (3.25):

$$\mu_{\Phi_i^{(1)}} = 1 - \frac{1}{W}, \tag{3.26}$$

i.e. the mean values of successive fire messages remain equidistant after the first firing cycle. The standard deviation of $\Phi_i^{(1)}$ is:

$$\sigma_{\mathrm{PCO}}^{(1)} = (1-\alpha)\sqrt{\sigma_{\Phi^{(0)}}^2 + \sigma_{\Delta}^2}. \tag{3.27}$$

Generalizing this to the $k$th phase update:

$$\begin{aligned}\Phi_i^{(k)} = (1-\alpha)^k \Phi_i^{(0)} + \sum_{j=1}^{k} (1-\alpha)^j \, \Delta_i^{(k-j)} \\ +\alpha\left(1-\tfrac{1}{W}\right)\textstyle\sum_{j=1}^{k}(1-\alpha)^j \pmod 1,\end{aligned} \tag{3.28}$$

with $\Delta_i^{(k-j)}$ iid random variables, each stemming from Definition 3. All $\Phi_i^{(k)}$ have equidistant mean values because of (3.26) and standard deviation given by (3.23). Similarly as for Proposition 1:

- each random variable $\Phi_i^{(k)}$ is a linear mixture of independent random variables
- $\sigma_{\mathrm{PCO}}^{(k)} > (1-\alpha)\sigma_{\Phi^{(0)}}$
- all densities $\mathrm{P}_{\Phi_i^{(k)}}$ have finite support since they are linear mixtures of PDFs with finite support.

Thus, via the central limit theorem [14] (see also Appendix 1 for more details), assuming sufficient updates take place, $\Phi_i^{(k)}$ will converge to normally-distributed random variables with equidistant mean values. We can thus reach convergence under Definition 4 for $k_{\mathrm{update}}$ given by (3.22). However, given that in PCO-based desynchronization the number of phase updates per firing cycle is not fixed, in order to derive the *expected number of firing cycles* until convergence is achieved, we need to derive the expected number of phase updates after each firing cycle. We can then match the expected number of phase updates until convergence to the expected number of firing cycles. The remainder of the proof is dedicated to this.

**Subsequent firing cycles, effect of neighboring firings:** A pictorial illustration of the PDF of $i$th firing, $\Phi_i$, during its $l$th phase update is given in Figure 3.4 in conjuction with its listening interval and the PDFs of $\Phi_{i-2}, \Phi_{i-1}$ and $\Phi_{i+1}$ (two previous and one subsequent firing). Since all phase random variables are normally distributed after a few phase updates, it is straightforward to infer from

Figure 3.4 that the probability that $\Phi_{i-1}$ will occur within $\Phi_i$'s listening interval is

$$\frac{1}{2}\mathrm{erf}\left(\frac{1}{W\sigma_{\mathrm{PCO}}^{(l-1)}\sqrt{2}}\right).$$

Moreover, the probability that $\Phi_{i-2}$ will occur within $\Phi_i$'s listening interval is

$$\frac{1}{2}\left[\mathrm{erf}\left(\frac{2}{W\sigma_{\mathrm{PCO}}^{(l-1)}\sqrt{2}}\right) - \mathrm{erf}\left(\frac{1}{W\sigma_{\mathrm{PCO}}^{(l-1)}\sqrt{2}}\right)\right].$$

This is also the probability that $\Phi_{i+1}$ (subsequent firing) will occur within the listening interval of $\Phi_i$.

**Subsequent firing cycles the effect of all firings within a window of $W$ firing events:** We can now generalize the previous calculation to the probability of occurrence of the $\left\lfloor\frac{W}{2}\right\rfloor$ firings $\Phi_{i-1}$ and $\Phi_{i+1}$ within $\Phi_i$'s listening interval. Beyond the $\Phi_{i-1}$, for the $j$th firing after the $\Phi_i$ or the $(j+1)$th firing before the $\Phi_i$ $(1 \leq j \leq \left\lfloor\frac{W}{2}\right\rfloor)$, this probability is

$$\frac{1}{2}\left[\mathrm{erf}\left(\frac{j+1}{W\sigma_{\mathrm{PCO}}^{(l-1)}\sqrt{2}}\right) - \mathrm{erf}\left(\frac{j}{W\sigma_{\mathrm{PCO}}^{(l-1)}\sqrt{2}}\right)\right].$$

Hence, summing up the probabilities of firing within $\Phi_i$'s listening interval for all $\left\lfloor\frac{W}{2}\right\rfloor$ firings before and after the $\Phi_i$, the expected number of phase updated during the $l$th firing cycle is given by (3.21). The total number of phase updates expected within $k$ firing cycles is:

$$\sum_{l=2}^{k} u_l + 1 - \frac{1}{W}.$$

As a result, for $k_{\mathrm{update}}$ phase updates leading to convergence under Definition 4 [shown by (3.22)], the corresponding number of firing cycles is given by (3.20). □

Proposition 2 shows that the PCO-based desynchronization, $k_{\mathrm{PCO}}$ is affected by $\alpha$, as well as by the noise assumptions, expressed by $\sigma_{\Phi^{(0)}}$ and $\sigma_{\Delta}$ in Definition 2 and Definition 3. The total number of firing, $W$, is also influencing the number of iterations for convergence to the steady state. However, as it will be shown by the next section (experiments), this effect is negligible in practice.

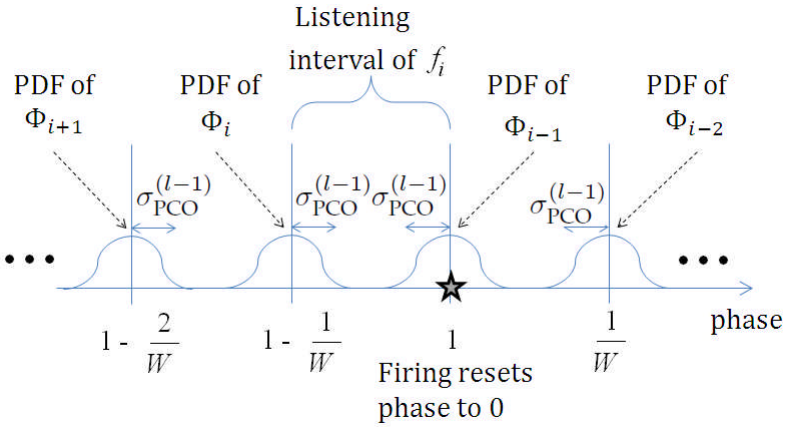


Figure 3.4: A pictorial illustration of the probability density functions of the phase random variables $\{\Phi_{i-2}, \Phi_{i-1}, \Phi_i, \Phi_{i+1}\}$ for the $l$th phase-update($l \geq 2$) of the $i$th firing via (3.5) .

## 3.4 Experimental Validation

For our experiments, we used the same hardware and settings as in the previous chapter, i.e. the iMote2 Crossbow WSNs with TinyOS 1.x. All nodes use the IEEE 802.15.4 standard with the default 2.4GHz Chipcon CC2420 wireless transceiver. We followed the TinyOS standard message format but reduced it to 2 data bytes when sending fire messages, since only the node number is required within a fire message. Similar to prior work [8], and as described in the previous chapter we reduced the backoff time to 1.2ms and, as described in Section 3.1 and 3.2, we used the local clock of each node to keep track of the node's own firing time as well as the firing times of the other nodes.

### 3.4.1 Conjecturing DESYNC and PCO as Second-order Systems

First we studied the experimental behavior of our system conjecturing it to be a second-order dynamic system, which is known to asymptotically converge with exponential rate to steady-state. If such a system is underdamped (e.g. due to the presence of noise), under a step input we expect to see an oscillatory response to the steady state. We compare the experimental response with a (noiseless) Matlab simulation of DESYNC and PCO as seen in Figure 3.5. It can be observed that the *noiseless* simulation model is close to a critically-damped system and thus converges to steady-state faster, while the experimental realization of DESYNC and

PCO indeed appears to resemble the response of a second-order underdamped system. This provides some intuitive justification to the exponential rates of convergence that have been conjectured in the literature but it also shows that such convergence rates may not in fact capture the actual convergence iterations required for DESYNC and PCO to reach the SS in a real deployment due to the underdamped nature of the system.

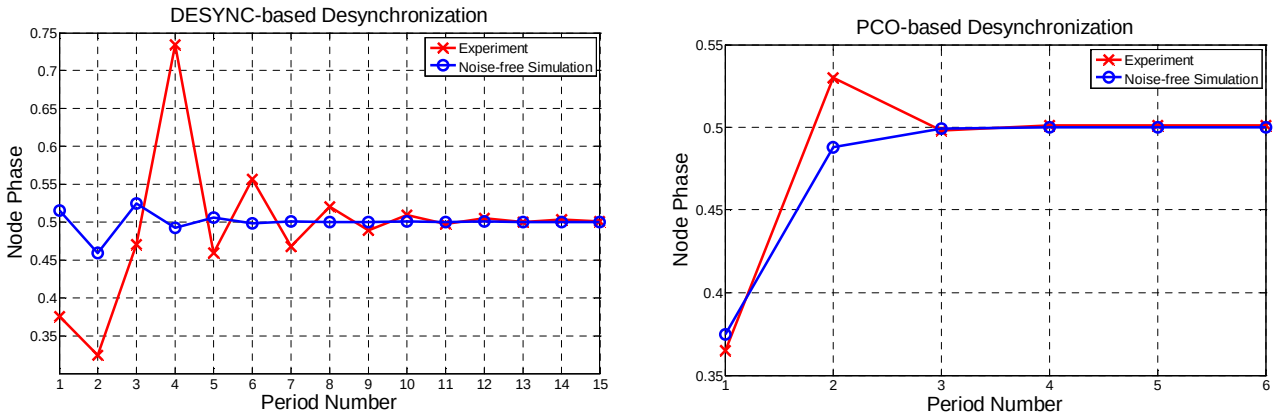


Figure 3.5. Node phase convergence to fixed phase for DESYNC (left) and PCO-based (right) approaches. The period number refers to the firing cycle based on the node's internal clock. "Simulation" is performed by Matlab in *noise-free conditions*, i.e. each phase is detected accurately and instantaneously by all nodes.

### 3.4.2 Standard Deviation of the Phase Measurement Noise

In the second part of our validation, we derived the standard deviation of the phase measurement noise, which can only be established via measurements from the real environment where DESYNC and PCO will be deployed.

The test environment was a standard University laboratory room, where interference from co-existing WiFi networks at the 2.4GHz range cannot be excluded. Our approach for measuring $\sigma_\Delta$ is as follows: *(i)* we implemented the DESYNC and PCO-based desynchronization in TinyOS nesC code as described in Section 3.1 and 3.2; *(ii)* we set $W = 4, 8, 16$ nodes and $\alpha = 0.95$ to ensure maximum coupling strength and $T = 1$s (which is the SS period value used in all our experiments) and *(iii)* we measured the oscillatory behavior of each node's phase after it has been left operating for prolonged interval of time to ensure

convergence to SS. The statistics of the oscillatory phase behavior observed via this experiment correspond to the marginal statistics of the phase during SS, i.e. the phase measurement noise accumulated due to interference and processing uncertainties. For both algorithms we found the standard deviation of the oscillating phase amplitude around the SS value of each node's phase to be $\sigma_\Delta = 0.34$ms and the accumulated phase statistics over all $W$ nodes were confirmed as marginally white. This was used for the validation of Proposition 1 and Proposition 2. No other parameter tuning is needed for the proposed model. This approach is easy to replicate under any real-world WSN setup.

### 3.4.3 Measurement and Simulation Setup

We are now ready to proceed with the experimental setup for both DESYNC and PCO-based desynchronization. In both cases, once all nodes were activated to transmit and receive on a single channel, a special "mix message" was broadcast by one of the nodes (chosen randomly) in order to trigger all nodes to set their initial fire message phase to correspond to a random interval within $T = 1$s from its reception. This creates the initial conditions of Definition 2. The nodes will then desynchronize their transmission of fire messages and converge to distributed TDMA. We present results under two convergence thresholds: $b_{\text{thres}} = 0.001$ and $b_{\text{thres}} = 0.02$ and with coupling coefficients $\alpha \in \{0.05, \dots, 0.95\}$. We use $c_{\text{conf}} = 1 - 10^{-4}$ to detect convergence under Definition 4 with near certainty. Under the experimental setup, each node detects convergence (or SS) by checking if (3.3) is valid for the last 5 firing cycles. After achieving SS and remaining in this state for 50 firing cycles, a node broadcasts another mix message, in order to repeat the process. This facilitates the automated collection of 100 experimental convergence iterations. Each node reported the number of firing cycles until convergence was detected (minus 4 cycles) via a special "report" message to a base station listening passively to all messages for monitoring purposes.

In order to cross-validate our theoretical and experimental results with simulations, we used the Matlab code of Degesys *et al* [8][145] for DESYNC and

added to it Matlab code for PCO with inhibitory coupling. We deliberately apply zero-mean additive noise in the phase update with $\sigma_\Delta = 0.34$ms and set each node to misfire with probability 0.4% in order to simulate the noise conditions observed in our experimental setup. Despite the fact that the simulation cannot capture the complex behavior of the real system in full detail, it allows for numerous desynchronization processes to be simulated (300 Matlab runs per triplet $\{W, \alpha, b_{\text{thres}}\}$ for each algorithm).

### 3.4.4 DESYNC Results

The results for this desynchroniztion mechanism are reported in Figure 3.6. All measurements around a value of α correspond to results with that value of $\alpha$; they have been plotted slightly separately solely for ease of illustration. For comparison purposes, we have also included the conjuecture of (3.4) for convergence to TDMA with DESYNC [8][13] in our results by scaling the order estimate to fit within the range of the obtained experiments and simulations. While we used $W = 16$ for this conjecture, this order of convergence has the same shape in function of $\alpha$ for other values of $W$ but simply needs different scaling to fit the range of the experiments. The results of Figure 3.6 show that the experiments and simulations appear to be independent of $W$, i.e. as expected from Proposition 1.

The WSN tends to converge to steady state faster when $\alpha$ decreases (until $\alpha$ = 0.25), since the presence of measurement noise causes higher-amplitude oscillations for strong coupling, i.e. for high values of $\alpha$. However, for very small values of $\alpha$, the convergence iterations increase dramatically due to weakened coupling between neighboring nodes. The model of Proposition 1 is within the standard deviation of the experimental results for all cases. In addition, the model prediction is within the standard deviation of the simulation results for the vast majority of cases, as seen in Figure 3.6.

Finally, by comparing the convergence results for low and high convergence threshold, one can observe that the use of small convergence threshold increases the converge iterations. The proposed model also predicted this behavior correctly

as shown by the figure of the Pearson correlation.

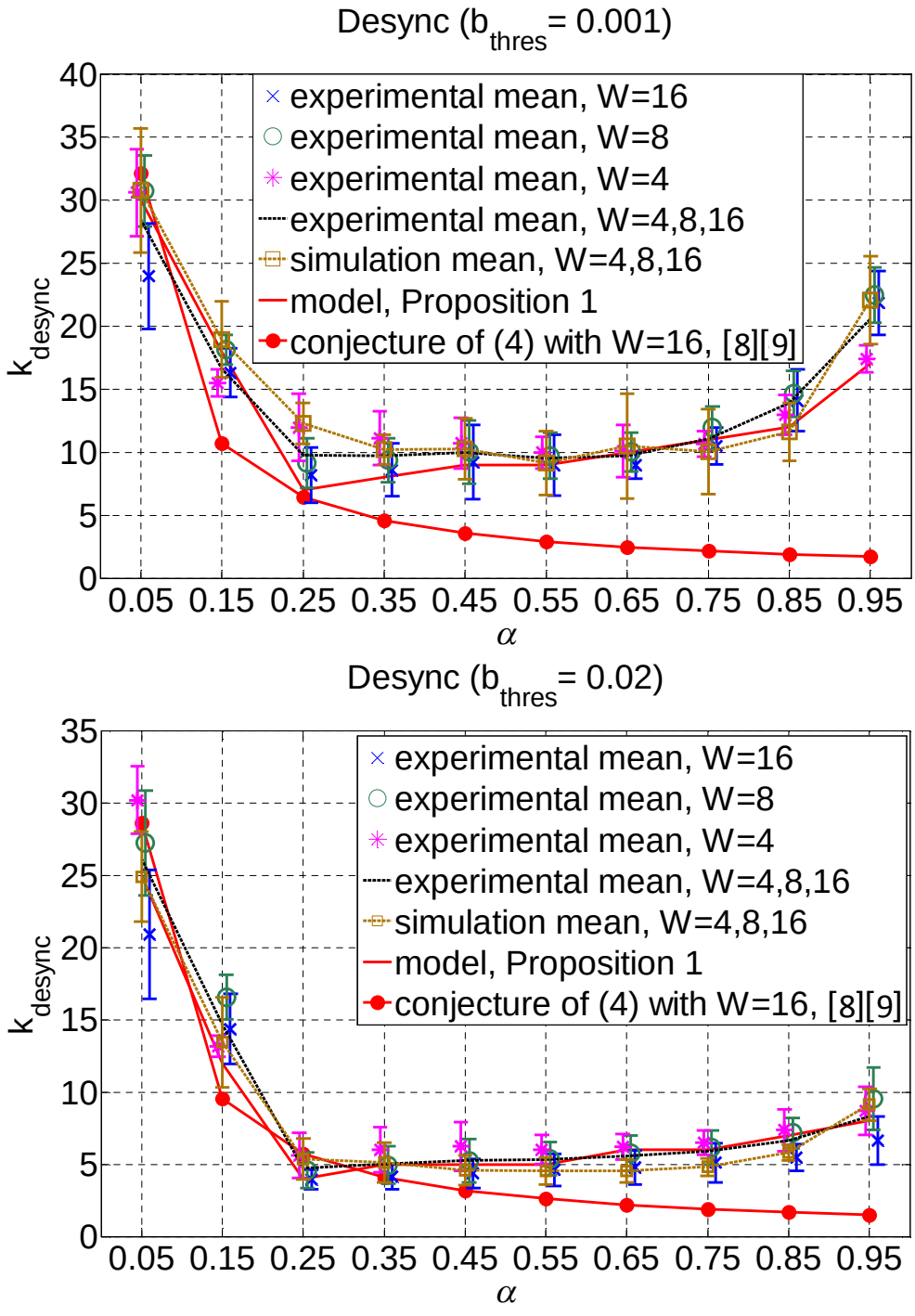


Figure 3.6: Required firing cycles for convergence for the DESYNC algorithm for various values of $\alpha$. The vertical error bars correspond to one standard deviation from the experimental (or simulation) mean values, which are indicated by marks.

For example the Pearson correlation coefficient between the simulation and experimental result is defined as the covariance of the two results divided by the product of their standard deviations. Therefore the Pearson correlation

coefficients for the simulation and the model curves against the mean experimental values were found to be: 0.9795 and 0.9723 (respectively) for $b_{\text{thres}} = 0.001$, while for $b_{\text{thres}} = 0.02$ they were 0.9939 and 0.9931 (respectively).

### 3.4.5 Results with PCO-based Desynchronization

The results are reported in Figure 3.7 for both small and large convergence thresholds. Our model exhibited no practical variation for different values of $W$; hence, for ease of illustration we present the proposed model with $W = 16$. Similarly, we used $W = 16$ for the lower bound of (3.6) in order to allow for the bound to be applicable for all values of $\alpha$ except of $\alpha \in \{0.85, 0.95\}$ [for which the constraint of $1 - \alpha > \frac{1}{W}$ of (3.6) is violated or is marginally applicable]. Since (3.6) derived negative estimates for most values of $\alpha$, we added an offset to the results of the bound to bring all of them to the non-negative region. Evidently, the bound of (3.6) does not match the observed behavior. We remark however that this is to be expected as the bound of (3.6) is derived under the assumption that each firing is influenced only by the firing of one neighboring node [9].

In this case, the system of nodes converges to SS faster for higher $\alpha$ values. Figure 3.7 demonstrates that the proposed model predicts this trend correctly and remains within one standard deviation from the experimental results and, for the majority of cases, within one standard deviation from simulation results. Since the model results do not change for different values of $W$, the firing events beyond the window of $W$ firing [(3.21) of Proposition 2] do not affect the model calculation; in other words, the inclusion of $W$ firing in (3.21) balances the results to the same values for different settings of $W$ tested, which agrees with the overall experimentally observed behavior of the system.

Finally, by comparing the convergence results for low and high convergence threshold, we note that the use of small convergence threshold increases the convergence iterations. The proposed model predicts this behavior correctly and agrees with the experimental trends reported. The Pearson correlation for the simulation and the model curves against the mean experimental values were:

0.9896 and 0.9739 (respectively) for $b_{\text{thres}} = 0.001$, while for $b_{\text{thres}} = 0.02$ they were 0.9977 and 0.9989 (respectively).

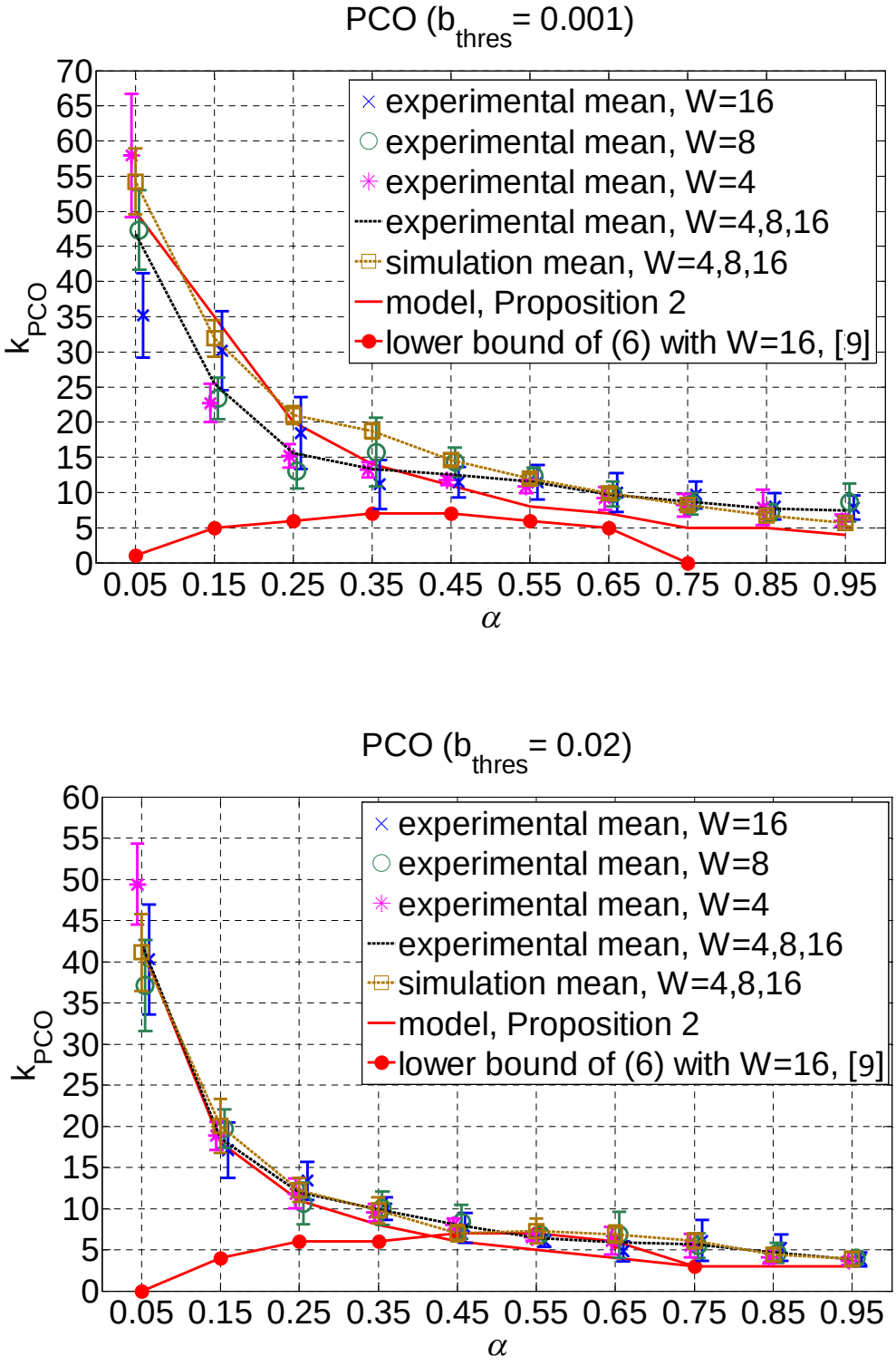


Figure 3.7: Required firing cycles for convergence for the PCO-based algorithm for various values of $\boldsymbol{\alpha}$. The vertical error bars correspond to one standard deviation from the experimental (or simulation) mean values, which are indicated by marks.

## 3.5 Discussion

By cross referencing between Figure 3.6 and Figure 3.7 we can compare the convergence iterations of both algorithms for different settings. Under appropriate choice of the coupling coefficient $\alpha$, the required firing cycles for convergence with PCO-based distributed desynchronization is comparable to the convergence cycles of DESYNC. As shown in Figure 3.6 and Figure 3.7, previous estimates or bounds that do not use a stochastic approach (and do not take into account the measurement noise conditions) are not a tight match to the experiments. To the best of our knowledge, this is the first work to derive stochastic convergence estimates for desynchronization and compare them against measurements from a real-world WSN deployment[2].

Our approach is also making a modeling simplification, which is discussed here. Both Proposition 1 and Proposition 2 make use of "stale" statistics for the stochastic characterization of convergence. Specifically, (3.15) and (3.23) as well as the description of the phase update process of (3.17), (3.25) and (3.28) assume that all random variables corresponding to the received fire messages have standard deviation corresponding to the previous phase update. However, each random variable corresponding to each phase variable of Definition 2 may have updated its standard deviation *during* each phase update. This is not taken into account when one assumes that all random variables have the same standard deviation for each phase update.

Concerning this point, we emphasize that, to calculate $\Phi_i^{(k)}$ and its moments for DESYNC and PCO-based convergence without making an explicit assumption on the exact firing order, one must assume the statistical moments for all phase variables of all firings are "stale" , i.e. they correspond to the previous phase update iteration, $k-1$. That is, it could be that a particular firing event is the first or the

[2] The only related attempt was found in [4]. However, that work proposes PCO with positive coupling for *synchronous* pulsing. The required differences (i.e. different phase update, reachback response, pre-emptive message staggering, RODL file [4]) do not permit a direct comparison of the convergence estimate of [4] with the experimental and theoretical results of this work.

last in a given phase update iteration. Since we do not assume any knowledge of this order (which may in fact not be fixed), Proposition 1 and Proposition 2 make use of the statistics of the *previous* phase update iteration for all nodes. Given that our stochastic modeling framework is in good agreement with the experimental and simulation results without requiring experimental tuning (besides knowledge of the standard deviation of the phase measurement noise), such a simplification can be considered as a good balance between the required assumptions and the modeling complexity.

In relation to WSN-based experiments of previous work [8],[13] that tend to use convergence thresholds that are an order of magnitude smaller and aim for TDMA systems with strict guard times, one can argue that the choice of a higher threshold for convergence, as seen in this chapter, can have an adverse effect on the achieved throughput once TDMA is reached since it leads to higher guard threshold. For example, Degesys *et al* [8] used similar WSN technology but, for increased TDMA accuracy, they managed to achieve 1ms threshold ($b_{\text{thres}} = 0.001$) in their experiments. However, we found that this does not provide for higher throughput in SS in comparison to setting $b_{\text{thres}} = 0.02$, i.e. as shown in this chapter. For example, the DESYNC and PCO algorithms for $W = 4$, $\alpha = 0.95$, $b_{\text{thres}} = 0.02$ and $T = 1$s lead to (approximately) 83kbps data throughput (which corresponds to 97% of the maximum throughput obtained between two nodes under the same environmental conditions). Similar results have been obtained for all other cases. Thus, that is why we would like to present results for $b_{\text{thres}} = 0.02$ that our results are comparable or superior to the ones reported in [Table 1, [8]]. Nevertheless the comparison of the proposed model with experiments, simulation results, and with previously known bounds for small guard threshold: $b_{\text{thres}} = 0.001$ (1ms for $T = 1$s) as proposed below:

Apparently, the experiment results for DESYNC-TDMA throughput and message loss is represented in Table 2.2 with the same $b_{\text{thres}}$ and $T$. To illustrate this, Table

3.2 and Table 3.3 present a comparison between the average bandwidth and convergence times obtained by using each of the two thresholds ($b_{\text{thres}}$ =0.02 and $b_{\text{thres}}$ =0.001) in DESYNC and PCO-based desynchronization for the indicative case of $W = \{4,8,16\}$ nodes and $\alpha = 0.75$. Following Degesys *et al* [8], the measured data throughput of all network nodes is normalized against the maximum data throughput between two iMote2 nodes under single sender-receiver communication (found to be 85.32kbps).

| **Threshold** | $\boldsymbol{b_{\text{thres}} = 0.02}$ | | | $\boldsymbol{b_{\text{thres}} = 0.001}$ | | |
|---|---|---|---|---|---|---|
| **DESYNC /Nodes** | **4** | **8** | **16** | **4** | **8** | **16** |
| Total Throughput (kbps) | 84.02 | 80.46 | 75.19 | 72.69 | 65.58 | 62.96 |
| Normalized, % | 98.48 | 94.30 | 88.13 | 85.19 | 76.86 | 73.79 |
| Max Individual (kbps) | 21.12 | 10.10 | 4.85 | 19.40 | 9.16 | 4.40 |
| Min Individual (kbps) | 20.86 | 9.93 | 4.40 | 16.81 | 7.50 | 3.36 |
| Convergence iterations | 6.5 [6] | 6.1 [6] | 5.1 [6] | 10.7 [11] | 11.9 [11] | 10.5 [11] |
| Message Loss (%) | 0.00 | 0.00 | 0.00 | 0.01 | 0.03 | 0.01 |

Table 3.2: DESYNC: average performance metrics under (guard) thresholds 20ms and 1ms. The numbers in brackets in the convergence iterations indicate the model prediction.

| **Threshold** | $\boldsymbol{b_{\text{thres}} = 0.02}$ | | | $\boldsymbol{b_{\text{thres}} = 0.001}$ | | |
|---|---|---|---|---|---|---|
| **PCO /Nodes** | **4** | **8** | **16** | **4** | **8** | **16** |
| Total Throughput (kbps) | 82.12 | 75.80 | 64.17 | 70.92 | 64.39 | 62.80 |
| Normalized, % | 96.26 | 88.84 | 75.21 | 83.12 | 75.47 | 73.60 |
| Max Individual (kbps) | 20.75 | 9.63 | 4.12 | 18.24 | 8.40 | 4.10 |
| Min Individual (kbps) | 20.11 | 9.26 | 3.88 | 17.26 | 6.54 | 3.35 |
| Convergence iterations | 5.5 [3] | 5.5 [3] | 6.1 [3] | 8.2 [5] | 8.4 [5] | 9.7 [5] |
| Message Loss (%) | 0.00 | 0.00 | 0.00 | 0.03 | 0.01 | 0.01 |

Table 3.3: PCO: average performance metrics under (guard) thresholds 20ms and 1ms. The numbers in brackets in the convergence iterations indicate the model prediction.

The tables illustrate that, when using $b_{\text{thres}} = 0.02$ (20ms), the results are superior to the ones with $b_{\text{thres}} = 0.001$ (1ms). This is because of the standard deviation of the marginal distribution of the phase measurement noise $\sigma_\Delta =$ 0.34ms (Subsection 3.4.2). This causes the nodes to occasionally lose their

converged (steady) state under $b_{\text{thres}} = 0.001$, thus requiring them to reapply DESYNC or PCO phase updates to converge again. Hence, although increasing the convergence threshold decreases the transmission slots' size per node, it can significantly enhance TDMA stability for DESYNC and PCO-based algorithms under real-world sensor network deployments, thereby leading to higher bandwidth.

Finally, the required firing iterations for convergence are decreased with increased convergence threshold, which makes the desynchronization algorithms more relevant to practical applications. The results of Table 3.2 and Table 3.3 show that the model remains close to the average experimental value and can be used as a predictor for the expected convergence time under a variety of settings.

## 3.6 Conclusion

Stochastic estimation of the convergence iterations to distributed time-division multiple access is proposed for two algorithms from the literature: the DESYNC approach and pulse-coupled oscillators with inhibitory coupling. Our stochastic estimates establish the expected firing cycles required until the probability density function of a node's phase falls within the *a-priori* determined threshold with very high confidence ($c_{\text{conf}} \times 100\%$ confidence of Definition 4). For both algorithms, our analytic expressions are validated successfully based on experiments with a fully-connected network of wireless sensor nodes under real-world conditions and based on simulations. Our model can estimate the convergence time under different configurations of desynchronization since it includes the influence of system parameters: total number of nodes, coupling coefficient, convergence threshold and phase measurement noise. As such, it can be used to estimate the best operational parameters (and associated delay) to establish distributed TDMA under several WSN protocols based the principle of desynchronization.

# Chapter 4
# Distributed Desynchronization In Multi-hop Wireless Sensor Networks

As shown in the previous two chapters, the DESYNC-TDMA algorithm works well for a fully-connected network and under single-channel operation. Nevertheless in a multi-hop network, each node cannot hear all other nodes of the system. Studies about the behaviour of the DESYNC on multi-hop networks have appeared in literature [15]; however, only simulations or theoretical analysis has been presented, without a real-world implementation. Degesys *et al* [15] consider the DESYNC on several topologies of WSNs: path graph, cycle graph, star graph, and unit-disk graph. Unlike [15], in this chapter we describe a modification of the DESYNC-TDMA algorithm to handle the multi-hop network scenario without considering specific topologies (Section 4.1).

## 4.1 Proposed Multi-hop DESYNC Protocol

Multi-hop desynchronization has been studied theoretically in [15][18]. The main issue with multi-hop network infrastructures is the "hidden terminal" problem. This problem is demonstrated in the topology of Figure 4.1, where, for example, broadcast fire messages from $n_5$ are only picked up by $n_1$. This hinters the synchronization of $n_1$, as $n_2, n_3, n_4$ are unaware that $n_5$ will interfere with their transmissions since that node is hidden from them.

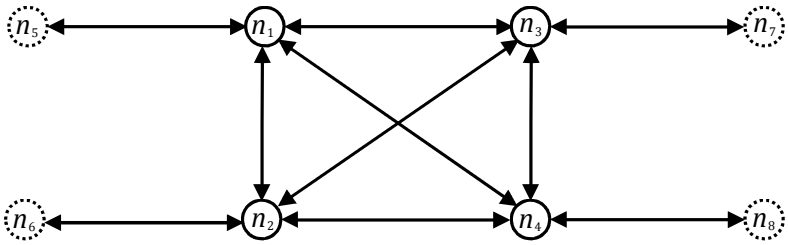


Figure 4.1: WSN with hidden nodes.

To address this issue, each node extends its own fire message to include the node numbers and the relative time instants that the node received their fire messages in relation to the time the node broadcasts its own fire message. Thus, reactive listening is extended to include the neighbours' information. This is achieved by each node maintaining a list of nodes from which fire messages have been received (direct fire messages) as well as a list of fire messages mentioned by other nodes (indirect fire messages). Nodes included in the second list but not in the first, are hidden nodes. When a node schedules its fire-message broadcast, it takes into account all other broadcasts inferred from both lists. In addition, the scheduling happens immediately after the node's own firing [and not as in Figure 2.13] and uses the previous period's fire message times.

### 4.1.1 Features and Implementation Details

In this section we describe the enhancements required for the multi-hop DESYNC in the way we implemented them. We start by identifying the format of the modified fire message, shown in Figure 4.2. The fire message for the multi-hop DESYNC includes the neighbour node ID (for all messages received by the sender node) and the difference time between the broadcast time of that message and the receipt of the neighbour fire message. The neighbour node ID is found from the sender ID in the broadcast fire message. These two values (neighbour node ID and difference time) are kept in the neighbour list, which includes the status of that neighbour node. The size of the neighbour ID is 1 byte per node. Similarly, the different time's length is 1 byte per node. In fact, the length of time variable in TinyOS has 4 bytes. It is converted from 4 bytes to 1 byte. Compacting the number of bytes needed for this information makes the fire message's length smaller, which leads to less overhead for its transmission and thereby better network performance. The fire message keeps the neighbour node ID and different time of up to 11 nodes as shown in Figure 4.2. This is the maximum number of nodes as we define in our experiments with multi-hop desynchronization, but it can be easily extended for larger WSNs if required.

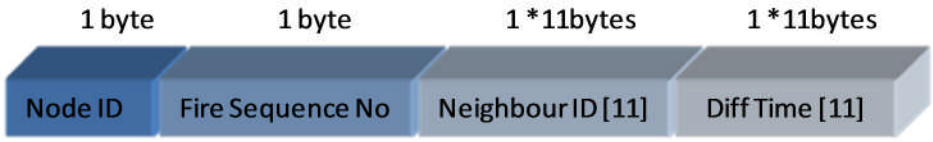


Figure 4.2: Format of a fire message of multi-hop DESYNC algorithm.

We explain how to find the hidden nodes using a broadcast fire message of any node in the code of Figure 4.3. Once the event of receiving a fire message is triggered, when getting the neighbour node ID from the fire message of multi-hop DESYNC, each node will check the status of every neighbour node ID in the fire message. By default, the node's status of all its neighbour is "HEARD". If the node's status is not "HEARD", that node will be a hidden node.

```
1  Event FireMessage.Received{
2     get neighborID
3     status_neighborID = HEARD
4     NumberNeighbor&Hidden++
5     For each neighbor ID in fire message
6     if (found termiantion)
7        return
8     else if (status_neighborID != HEARD)
9        status_neighborID = HIDDEN
10       NumberNeighbor&Hidden++
11 }
```

Figure 4.3: Code for multi-hop DESYNC version that the hidden node(s).

For a multi hop network, the timing information in the fire message is used for the scheduling of the next fire time, $t_i^{(k)}$. This algorithm needs to find the new *NextNode FireTime,* $t_{i+1}^{(k-1)}$ and *LastNode FireTime,* $t_{i-1}^{(k-1)}$. This is achieved using the neighbour ID and difference times in the fire messages. When a node receives the fire message, it will collect the node ID, status and the relative time of any neighbour node into the table list. Then the node will define the $t_{i+1}^{(k-1)}$ and $t_{i-1}^{(k-1)}$before calculating the $t_i^{(k)}$. Specifically, it finds the two neighbour nodes considering also potential hidden nodes; therefore the node needs to check the difference time between its own fire time and other's fire time for two cases which are in the past time, $k-2$ and at the current period, $k-1$ as demonstrated in Figure 4.4 . When considering the minimum of the difference time, it can be the

time at $t_{i-1}^{(k-2)}$and $t_{i+1}^{(k-2)}$ which will be set as the $t_{i-1}^{(k-1)}$ and $t_{i+1}^{(k-1)}$ respectively in the case I. On the other hand, the node can hear the fire message at the $t_{i-1}^{(k-1)}$, not the $t_{i-1}^{(k-2)}$ That means this time will be the $t_{i-1}^{(k-1)}$. Therefore this is case II which the $t_{i+1}^{(k-1)}$ is calculated from the fire time at the $t_{i+1}^{(k-2)}$ by adding a period ($T$).

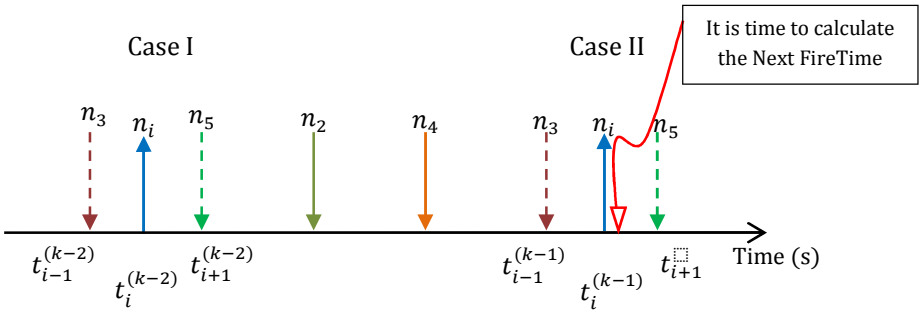


Figure 4.4: Local view for multi-hop DESYNC to find the $t_{i+1}^{(k-1)}$ and $t_{i-1}^{(k-1)}$.

In order to obtain the smallest of the difference time, the node has to check through all neighbour node list and the hidden node list to determine which is the last node and next node are. First of all, node examines the neighbour node list as shown the code in Figure 4.5. The node needs to initialize the *min_diff_prev* and *min_diff_next* with a large value. Then it will compare these two values with the difference time, which is the time between its own fire time and any other node's fire time. As described previously, we need to consider this comparison both for the $(k-2)$ period and for the $(k-1)$ period, in order for the node has to discover the new $t_{i+1}^{(k-1)}$ and $t_{i-1}^{(k-1)}$ from the neighbour and hidden node list correctly.

```
1  For all neighbor nodes except me
2     diff_pastTime = NeighborTime - past_FireTime
3     diff_currTime = NeighborTime - current_FireTime
4     if (diff_pastTime < 0)
5        if (-diff_pastTime < min_diff_prev)
6           min_diff_prev = -diff_pastTime
7           LastNode_FireTime = NeighborTime
8           prevTime_pastFire = TRUE
9     else
10       if (diff_pastTime < min_diff_next)
11          min_diff_next = diff_pastTime
12          NextNode_FireTime = NeighborTime
13
14    if (-diff_currTime < min_diff_prev)
15       min_diff_prev = -diff_currTime
16       LastNode_FireTime = NeighborTime
17       prevTime_pastFire = FALSE
```

Figure 4.5: Code for multi-hop DESYNC algorithm when checking the neighbour list.

After checking neighbour list, the node continues to examine the hidden node list with the same procedure, as shown in Figure 4.6.

```
For all hidden nodes
   diff_pastTime = HiddenTime - past_FireTime
   diff_currTime = HiddenTime - current_FireTime
   if (diff_pastTime < 0)
      if (-diff_pastTime < min_diff_prev)
         min_diff_prev = -diff_pastTime
         LastNode_FireTime = HiddenTime
         prevTime_pastFire = TRUE
   else
      if (diff_pastTime < min_diff_next)
         min_diff_next = diff_pastTime
         NextNode_FireTime = HiddenTime

   if (-diff_currTime < min_diff_prev)
      min_diff_prev = -diff_currTime
      LastNode_FireTime = HiddenTime
      prevTime_pastFire = FALSE
```

Figure 4.6: Code for multi-hop DESYNC when checking the hidden node list.

When we have finished these procedures, we can calculate the $t_i^{(k)}$ for two cases as shown in Figure 4.4 . We consider the *prevTime_pastFire* status for selecting the criteria. If it is TRUE, this means we compute the time $t_i^{(k)}$ according to the period $k-2$ (case I):

$$t_i^{(k)} = T + (1-\alpha)t_i^{(k-1)} + \alpha\left(\frac{t_{i-1}^{(k-2)} + t_{i+1}^{(k-2)}}{2} + T\right) \tag{4.1}$$

On the other hand, the case is *prevTime_pastFire* = FALSE, node will schedule the $t_i^{(k)}$ based on the interval $k-1$ (case II) by:

$$t_i^{(k)} = T + (1-\alpha)t_i^{(k-1)} + \alpha\frac{t_{i-1}^{(k-1)} + t_{i+1}^{(k-2)} + T}{2} \tag{4.2}$$

## 4.2 Experimental Results

We tested a multi-hop DESYNC network with 8 motes and used the topology of Figure 4.1 as an example. We run five tests of 60 seconds for each case: when using the proposed algorithm for discovering and incorporating hidden nodes and when using the conventional DESYNC algorithm of Chapter 2. It can be observed from Figure 4.1 that $n_1$ is linked to $n_5$, $n_2$ is linked to $n_6$, $n_3$ is linked to $n_7$ and $n_4$ is linked to $n_8$. Hence, $n_5 \sim n_8$ are all hidden nodes. When testing with the hidden-node

implementation of the fire message broadcasts, we validated that each hidden node can discover all fire messages from all 8 nodes. For example, node 5 is a hidden node that discovers all fire messages from reading the fire message of node 1. For the conventional DESYNC, i.e. without the hidden node implementation, each hidden node can hear only the fire message from the node it is directly linked to. For instance, node 5 only discovers node 1, whereas node 1 can hear fire messages from node 2, 3, 4 and 5.

In the first round of experiments, we set a single-channel mode and arranged the topology of Figure 4.1 using UCL's anechoic chamber for antenna measurements[3] to create an interference-free test environment that also includes obstacles with absorbing material limiting the broadcast messages of each node to the indicated links in the topology. This allows for testing of the multi-hop desynchronization of Section 4.1 against the conventional single-hop DESYNC scheme [8].

The average throughputs after 10 tests of 60s each are given in Table 4.1 against conventional (1-hop) DESYNC [8]. It is particularly evident from the results that the inclusion of the neighbours' node list alleviates the hidden-node problem and maintains high throughput and low message loss for the proposed multi-hop desynchronization approach.

| **Scheme** | **Proposed Multi-hop** | **1-hop [8]** |
|---|---|---|
| Total throughput (kbps) | 68.7 | 27.6 |
| Normalized, % | 80.5 | 32.3 |
| Max per node (kbps) | 9.6 | 6.1 |
| Min per node (kbps) | 7.9 | 1.6 |
| Message loss (%) | 0.1 | 1.0 |

Table 4.1: Results under the multi-hop topology of Figure 4.1. The presented measurements include the period after SS has been obtained.

For 8 nodes, we found the total throughput of the proposed multi-hop extension decreased by approximately 15% in comparison to the conventional DESYNC design

[3] http://www.ee.ucl.ac.uk/about/anechoic-chamber

under the fully-connected topology. For the partly-connected topology, the total throughput of conventional DESYNC without the hidden node decreased by approximately 60% from the 68.7 kbps obtained by the proposed multi-hop DESYNC. The message loss also gets to approximately 1%. For the multi-hop DESYNC, we set up the new threshold from 10 ms to 15 ms and *time_SendData_onePack* from 8 ms to 15 ms in order to make the system stable.

## 4.3 Conclusion

By extending the primitive of distributed desynchronization in wireless networks, we proposed (and demonstrated) the improvements offered by including neighbour-node information to avoid the hidden node problem in multi-hop topologies. Experimentation using TinyOS iMote2 wireless sensors demonstrated the throughput increase provided by our proposed multi-hop mechanism and validated our theoretical findings.

# Chapter 5
# Distributed Time-Frequency Desynchronization In Wireless Sensor Networks

In this chapter we study the capability of desynchronization-based communications via multiple channels. The proposed multichannel protocol improves the throughput performance against single-channel desynchronization-based WSNs [16].

Complementary to desynchronization for distributed TDMA, multi-channel MAC protocols aim for load balancing via frequency division multiple access [37]-[39], or TDMA combined with pseudo-random channel hopping, e.g. as proposed for the upcoming IEEE 802.15.4e standard [29]. The key principles are: *(i)* collection of traffic statistics or TDMA coordination by a central station; *(ii)* centralized TDMA and channel assignment (or hopping) for interference reduction.

In this chapter, we propose distributed MAC-layer time-frequency division multiple access (TFDMA) for WSNs based on reactive listening of message broadcasts. Unlike previous TFDMA schemes [37]-[39] that are centralized or highly-complex for real-world sensor devices (due to complex heuristics or NP-time algorithms), our approach forms a low-complex decentralized scheme based on reactive listening [105]. In addition, unlike channel hopping approaches based on IEEE 802.15.4e MAC [29][30], we avoid continuous channel switching; we instead provide for distributed TDMA and channel assignment that is compatible with the widely-supported IEEE 802.15.4 MAC and is also applicable within other multichannel MAC protocols without requiring explicit hardware support.

Beyond the proposed TFDMA, this chapter's contributions are: *(i)* we prove that distributed TFDMA converges to steady state under appropriate parameter settings; *(ii)* we derive the expected delay for convergence to SS; *(iii)* we perform

real-world validation of the proposed scheme via TinyOS iMote2 nodes and make our source code available online [45].

## 5.1 Proposed Multi-channel Extension

Standards suitable for wireless sensors, such as the IEEE 802.15.4 MAC, allow for half-duplex communications over a selection of channels at 2.4GHz with minimal cross-channel interference. This hints that, should TDMA desynchronization be extended to $C$ channels ($C > 1$), increased throughput per node will be observed since $[\![W_{\text{tot}}/C \pm 0.4\bar{9}]\!]$ nodes [4] will operate in each channel. The highest throughput can be achieved when the number of nodes is balanced in all channels [37]. For example, for $C = 2$, the aim would be to "spontaneously" separate $W_{\text{tot}} = 8$ nodes into two distinct sets: $W_1 = W_2 = 4$, i.e. 4 nodes in each channel. This uses the allocated spectrum of IEEE802.15.4 twice as efficiently in comparison to PCO-based TDMA [8][9]. However, channel switching must be designed judiciously, as frequent channel switching causes loss of (de)synchronization due to variable hardware and operating system latencies and additional effort (and energy consumption) is required to recover it [44].

### 5.1.1 Proposed Protocol

TFDMA extends the reactive listening primitive and makes for a stable process for time-frequency node balancing. By utilizing reactive listening, it only allows for channel switching if less nodes are detected in the new channel. The detailed operation is described here.

**Switching:** In the beginning, each wireless sensor picks a channel Ch$\{c\}$ ($1 \leq c \leq C$) randomly after that applies DESYNC [8]. After $k_{\text{ss}}$ periods, convergence to TDMA is achieved [via the check of the convergence state ( $\left|t_i^{(k_{\text{ss}}+1)} - t_i^{(k_{\text{ss}})} - T\right| < b_{\text{thres}}T$)]. We present the scenario which sensor nodes join in the random channel as in Figure 5.1, this means that, on average, TFDMA will begin in a near balanced state. We have proposed our delay analysis to reflect this, seen by Proposition 2.2,

[4] $W_{\text{tot}}$ indicates the total number of nodes and $W_c$ represents the number of nodes operating in channel $c$ ($Ch\{c\}$); ⌊a⌋, ⌈a⌉ and ⟦a⟧ are the floor, ceiling and round operations.

which now does the averaging of the expected delay under each possible initial state, multiplied by the probability that this initial state will happen when $W_{\text{tot}}$ nodes join $C$ channels randomly.

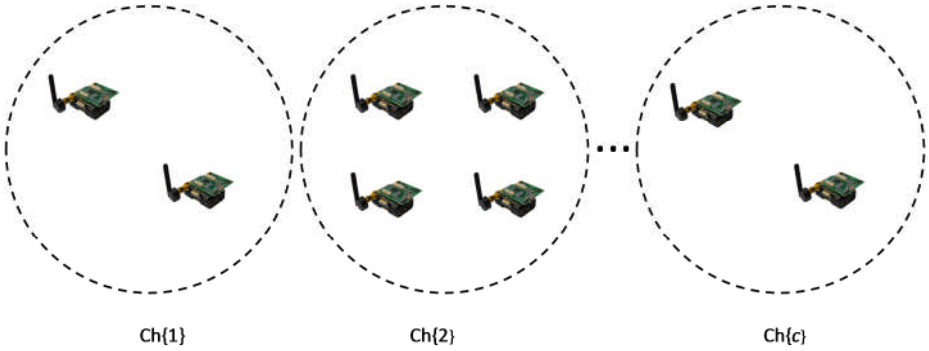


Figure 5.1: A sample of the random channel selection case scenario of the proposed multi-channel extension.

After $k_{\text{ss}}$ periods, convergence to TDMA is achieved [via the check of $\left|t_i^{(k_{\text{ss}}+1)} - t_i^{(k_{\text{ss}})} - T\right| < b_{\text{thres}}T$ with $b_{\text{thres}}$ a preset threshold, e.g. $b_{\text{thres}} = 0.02$]. Subsequently, after broadcasting its fire message, each node can switch to the previous or next channel, i.e. from $\text{Ch}\{c\}$ to $\text{Ch}\{c + s_c\}$ $(1 \le c \le C$, with $s_c \in \{\pm 1, \dots, \pm\lfloor C/2 \rfloor\}$ and cyclic extension: $\text{Ch}\{C + |s_c|\} \equiv \text{Ch}\{|s_c|\}$, $\text{Ch}\{1 - |s_c|\} \equiv \text{Ch}\{C + 1 - |s_c|\}$), by broadcasting a "switch" message in $\text{Ch}\{c\}$. This message contains the node number and alerts all other nodes listening and transmitting in $\text{Ch}\{c\}$ that this node will attempt to switch to a different channel. Once receiving one switch message, all other nodes in $\text{Ch}\{c\}$ disable the desynchronization process and, instead of assigning their next fire-message broadcast based on $t_i^{(k+1)} = T + (1-\alpha)t_i^{(k)} + \alpha\frac{t_{i-1}^{(k)} + t_{i+1}^{(k)}}{2}$, they simply repeat it after $T$s for the next period. This is termed "switch" mode.

**Reactive listening:** The node attempting to switch to $\text{Ch}\{c + s_c\}$ listens to the fire messages of $\text{Ch}\{c + s_c\}$ for one period[5] and determines if $W_{c+s_c} \le W_c - 2$. If so, it joins the new channel and distributed TDMA is achieved in $\text{Ch}\{c\}$ and $\text{Ch}\{c + s_c\}$ via DESYNC. Otherwise it returns to $\text{Ch}\{c\}$, broadcasts a "return" message, and

[5] Each beacon message includes the total number of nodes heard in $\text{Ch}\{c\}$, as well as a flag indicating whether the channel is in switch mode (i.e. whether a node has left to listen to $\text{Ch}\{c + s_c\}$). Thus, each node finds $W_c$ (and whether switch mode is on) even if only a single beacon message is heard in $\text{Ch}\{c\}$.

rejoins desynchronization and data transmission in Ch$\{c\}$. Nodes in Ch$\{c\}$ exit the switch mode and continue their regular desynchronization operation when a return message is received, or after two periods.

Assuming $s_c^{(k)} > 0$ for the $k$th switch mode of Ch$\{c\}$, if a return message is received, all nodes in Ch$\{c\}$ set $s_c^{(k+1)} = -s_c^{(k)}$, i.e., when unsuccessful, the switching direction changes; furthermore $s_c$ gradually increases up to $\pm\lfloor C/2 \rfloor$ to cover all channels. An update occurring simultaneously between channels: $c \to c + s_c^{(k)}$ and $\acute{c} \to c$ ($1 \le \acute{c} \le C$ & $\acute{c} \ne c$) is expressed stochastically for Ch$\{c\}$ by:

$$\begin{aligned} \bar{W}_c^{(k+1)} = \bar{W}_c^{(k)} &- \min\left\{u\left[\bar{W}_c^{(k)} - 2 - \bar{W}_{c+s_c^{(k)}}^{(k)}\right] p_{\mathrm{sw},c}^{(k)} \bar{W}_c^{(k)}, 1\right\} \\ &+ \min\left\{u\left[\bar{W}_{\acute{c}}^{(k)} - 2 - \bar{W}_c^{(k)}\right] p_{\mathrm{sw},\acute{c}}^{(k)} \bar{W}_{\acute{c}}^{(k)}, 1\right\} \end{aligned} \tag{5.1}$$

with: $\bar{W}_c^{(k)}$ the expected number of nodes at Ch$\{c\}$ after the $k$th switch mode; $u[\cdot]$ the unit-step function, used to identify whether switching can occur between channels $c \to c + s_c^{(k)}$ and $\acute{c} \to c$; and $p_{\mathrm{sw},c}^{(k)}$, $p_{\mathrm{sw},\acute{c}}^{(k)}$ the switching probabilities of a node in Ch$\{c\}$ and Ch$\{\acute{c}\}$.

**Stability and convergence mechanism:** Since each node decides and sends its switch message immediately after its fire message, once one such message is heard in one period, the remaining nodes in that channel cannot switch in this period. The switch mode allows for undisturbed operation while nodes find out if the previous or next channel has less nodes: *(i)* if a node returns, it can quickly regain its previous TDMA slot with minimal disturbance; *(ii)* via the switch mode, the reactive listening primitive of (5.1) is used for adjustment of the number of nodes per channel. Once the switch mode is exited for the $k$th time in Ch$\{c\}$, each node modifies its switching probability by:

$$p_{\mathrm{sw},c}^{(k)} = \min\left\{\beta^v \times p_{\mathrm{sw},c}^{(k-1)}, 1\right\} \tag{5.2}$$

where: $v = 1$ if no return message is received, $v = -1$ otherwise, and $\beta > 1$. Notice that $p_{\mathrm{sw},c}^{(0)}$ controls how quickly the nodes of channel Ch$\{c\}$ will attempt to switch initially and $\beta$ controls the "back-off" from switching (also preset), and $v$ changes according to the result of the last switch attempt.

Notice that, once $[\![W_{\text{tot}}/C \pm 0.4\bar{9}]\!]$ nodes exist in all channels, further switching attempts will cause the nodes to return to their original channel, thus leading to $\forall c: p_{\text{sw},c}^{(\text{ss})} \to 0$ from (5.2). Thus, even in steady state we enforce infrequent channel switching attempts to periodically discover and compensate for potential imbalances created by nodes departing unexpectedly (e.g. if nodes malfunction): we impose that a node in each channel will attempt to switch after $Z$ periods of switching inactivity.

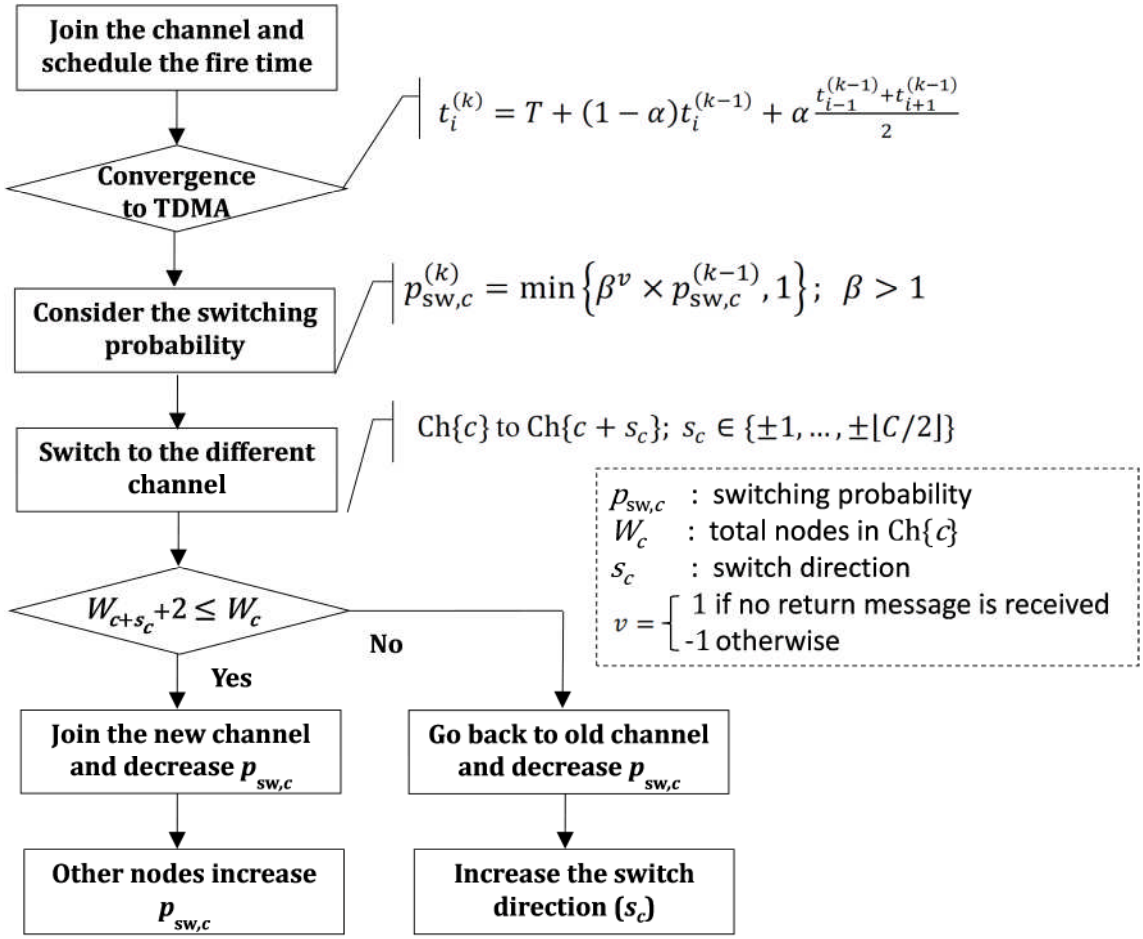


Figure 5.2: The diagram of the proposed multi-channel extension

Both the periodic fire message broadcasts and the reactive listening principle are of critical importance for (5.1) and for the proposed TFDMA operation as they ensure switching nodes can detect the number of nodes in the new channel (and whether the new channel is in fact in switch mode).

The proposed protocol has two tunable parameters[6]: $p_{\mathrm{sw},c}^{(0)}$ and $\beta$. They are linked together via the update of $p_{\mathrm{sw},c}^{(k)}$ given by eq. (5.2). Notice that the original DESYNC anyway had two tunable parameters: $b_{\mathrm{thres}}$ and $\alpha$. The exploration of the specific effect of these four parameters on the convergence delay remains a topic for future work. Given we provide our full source code in the [45], comparisons under different settings for $p_{\mathrm{sw},c}^{(0)}$ and $\beta$ and $b_{\mathrm{thres}}$ and $\alpha$ are easy to perform by any experimentalist in this area.

### 5.1.2 Properties

***Proposition 1 (Convergence to SS):*** An arbitrary distribution of $W_{\mathrm{tot}}$ nodes in $C$ channels ($W_{\mathrm{tot}} \geq 2C$) will be driven to balanced state of $[\![W_{\mathrm{tot}}/C \pm 0.4\bar{9}]\!]$ nodes per channel under TFDMA with $0 < \alpha < 1$.

*Proof:* Single-channel TDMA desynchronization has already been shown to converge for $0 < \alpha < 1$ [8][9]. Thus, it suffices to show that the proposed channel switching mechanism leads to balanced number of nodes per channel.

It is straightforward to check that the vectors comprising $W_c^{(\mathrm{ss})} = [\![W_{\mathrm{tot}}/C \pm 0.4\bar{9}]\!]$, $\forall c: 1 \leq c \leq C$, are eigenvectors (with unity eigenvalue) of the matrix system formed for all channels via (5.1). Thus $W_c^{(\mathrm{ss})}$ are the fixed points of the system of (5.1). For every iteration $k$, the matrix system formed by (5.1) for all $C$ channels has all its eigenvalues on or within the unit circle. Limit cycles are avoided as u[·] ensures updates will happen only when leading to balanced number of nodes. Thus, $\forall c: \lim_{k\to\infty} W_c^{(k)} = W_c^{(\mathrm{ss})}$.

Assuming that for every channel $c$ $(1 \leq c \leq C)$: $s_c^{(k)} = 1$, then the transition system formed by [(5.3),[43]] for all $C$ channels is written in matrix form as:

$$\bar{\mathbf{w}}^{(k+1)} = \mathbf{G}^{(k)}\, \bar{\mathbf{w}}^{(k)} \tag{5.3}$$

with $$\bar{\mathbf{w}}^{(k+1)} = \left[\bar{W}_1^{(k+1)} \quad \bar{W}_2^{(k+1)} \quad \cdots \quad \bar{W}_{C-1}^{(k+1)} \quad \bar{W}_C^{(k+1)}\right]^T \tag{5.4}$$

[6] There is also parameter $Z$ which forces a switching attempt and can be set to any number of periods higher than the number of periods required for convergence to steady state. In practice, this is simply a periodic "nudge" of the system in order consider random unbalances occurring from nodes disappearing because they terminated their communications unexpectedly (e.g. due to malfunction or low battery).

$$\bar{\mathbf{w}}^{(k)} = \begin{bmatrix} \bar{W}_1^{(k)} & \bar{W}_2^{(k)} & \cdots & \bar{W}_{C-1}^{(k)} & \bar{W}_C^{(k)} \end{bmatrix}^T \tag{5.5}$$

$$\mathbf{G}^{(k)} = \begin{bmatrix} 1-g_1 & 0 & \cdots & 0 & g_C \\ g_1 & 1-g_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1-g_{C-1} & 0 \\ 0 & 0 & \cdots & g_{C-1} & 1-g_C \end{bmatrix} \tag{5.6}$$

and $\forall c: g_c = u\left[\bar{W}_c^{(k)} - \bar{W}_{c+1}^{(k)} - 2\right] p_{\mathrm{sw},c}^{(k)}$ with the constraint of $g_c \bar{W}_c^{(k)} \leq 1$ due to the min{·} operators of [(5.3),[43]], i.e. only one node is allowed to switch at any given moment.

For the general case of $s_c^{(k)} \neq 1$, factors $g_c$ of **G** are positioned in column $c$ and row $c + s_c^{(k)}$, with cyclic extension at the borders (i.e. when $c + s_c^{(k)} > C$ or $c + s_c^{(k)} < 1$). The stochastic transition matrix **G** of (5.3) under any $s_c^{(k)}$ is a left-stochastic matrix with: its columns maximally summing to unity, all its entries being non-negative and each entry is smaller or equal to unity. As such, via the Perron–Frobenius theorem [43], we identify that the maximum magnitude of all eigenvalues of **G** is unity, i.e. all eigenvalues of any instantiation of **G** are within the unit circle. Hence, under iterations with stochastic matrices **G**, the system of (5.3) will converge to a steady state or to a limit cycle. Limit cycles, i.e. oscillations between unbalanced number of nodes per channel, are avoided since, under the reactive listening [expressed stochastically by (5.1)], nodes switch only if they join a channel with less nodes. The inclusion of the total number of nodes (and switch mode status) of each channel with *each* beacon message[7] ensures that no erroneous node switching can occur during convergence to SS even under the occasional loss of a switch or beacon message. Hence, the system of (5.3) will converge to a steady state. All vectors:

$$\mathbf{w}^{(\mathrm{ss})} = \begin{bmatrix} [\![W_{\mathrm{tot}}/C \pm 0.4\bar{9}]\!] & \cdots & [\![W_{\mathrm{tot}}/C \pm 0.4\bar{9}]\!] \end{bmatrix}^T \tag{5.7}$$

[7] Each beacon message includes the total number of nodes heard in Ch$\{c\}$, as well as a flag indicating whether the channel is in switch mode (i.e. whether a node has left to listen to Ch$\{c + s_c\}$). Thus, each node finds $W_c$ (and whether switch mode is on) even if only a single beacon message is heard in Ch$\{c\}$.

comprise the eigenvectors (fixed points) of the system of (5.3) with $\mathbf{G} = \mathbf{I}$ [i.e. all eigenvalues corresponding to the eigenvectors of (5.7) are 1] since all $\mathrm{w}^{(\mathrm{ss})}$ of (5.7) lead to:

$$\forall x, y \in \{1, \dots, C\} : \max\left\{\left|\bar{W}_x^{(k)} - \bar{W}_y^{(k)}\right|\right\} = 1$$
$$\Rightarrow \forall x, y: u\left[\bar{W}_x^{(k)} - \bar{W}_y^{(k)} - 2\right] = u\left[\bar{W}_y^{(k)} - \bar{W}_x^{(k)} - 2\right] = 0$$
$$\Rightarrow \forall c: g_c = 0.$$

Thus:

$$\forall c: \lim_{k\to\infty} W_c^{(k)} = [\![W_{\mathrm{tot}}/C \pm 0.4\bar{9}]\!]. \tag{5.8}$$

Since $\mathbf{G} = \mathbf{I}$ for all fixed points, limit cycles are avoided once reaching one of the $\mathbf{w}^{(\mathrm{ss})}$ of (5.7). Also, the system of (5.3) is guaranteed to converge to one of the eigenvectors of (5.7) since only single-node transitions occur. This avoids limit cycles between vectors:

$$\mathbf{w}^{(\mathrm{lc})} = [[\![W_{\mathrm{tot}}/C \pm 0.\bar{9}]\!] \quad \cdots \quad [\![W_{\mathrm{tot}}/C \pm 0.\bar{9}]\!]]^T \tag{5.9}$$

or between other combinations of unbalanced number of nodes amongst $C$ channels. □

Since nodes join channels randomly, there is no ranking specified: each node will fire according to DESYNC and will listen to other firings for timing synchronization and for channel distribution. Thus, even if a node gets a chance to switch, new nodes come (or leave) from its channel**.** Hence, after the converged state, a balanced number of nodes will exist in each channel. Essentially, it is the same as a lottery: whether a node stays in the same channel or whether it moves to a different channel, it will reach a balanced state of $[\![W_{\mathrm{tot}}/C \pm 0.4\bar{9}]\!]$ nodes per channel.

***Proposition 2 (Expected Delay until Convergence to SS)*****:** For TFDMA with $W_{\mathrm{tot}}$ nodes in $C$ channels, the expected delay (in s) until convergence to balanced state can be estimated by:

$$d_{W_{\mathrm{tot}},C} = T\left[\sum_{i=1}^{\frac{(W_{\mathrm{tot}}+C-1)!}{(C-1)!W_{\mathrm{tot}}!}}\left[p(i)\sum_{k=1}^{W_{\mathrm{diff}}(i)}\left(d^{(k)}+2\right)\right] + k_{\mathrm{ss}}\right] \tag{5.10}$$

with: $i$ the index of the vector comprising a possible distribution of $W_{\mathrm{tot}}$ nodes in $C$ channels (i.e. $[W_1(i) \dots W_C(i)]$,

$$p(i) = \prod_{c=1}^{C-1}\left[\binom{W_{\text{res},c}(i)}{W_c(i)}\frac{(C-1)^{W_{\text{res},c}(i)-W_c(i)}}{C^{W_{\text{res},c}(i)}}\right] \tag{5.11}$$

$$\text{and } d^{(k)} = \frac{1-\left(1-\beta^{k-1}p_{\text{sw},c}^{(0)}\right)^{Z\left(W_{\text{diff}}(i)+[\![W_{\text{tot}}/C]\!]-k+1\right)}}{1-\left(1-\beta^{k-1}p_{\text{sw},c}^{(0)}\right)^{W_{\text{diff}}(i)+[\![W_{\text{tot}}/C]\!]-k+1}} \tag{5.12}$$

with $\forall i$: $W_{\text{res},c}(i) = W_{\text{tot}} - \sum_{m=1}^{c-1} W_m(i)$, and

$$W_{\text{diff}}(i) = \max_{\forall c}|W_c(i) - [\![W_{\text{tot}}/C]\!]|. \tag{5.13}$$

*Proof:* This proposition shows the influence of design settings: $p_{\text{sw},c}^{(0)}$, $\beta$ and $k_{\text{ss}}$ (controlled by $\alpha$ [8]), as well as system parameters: $C$, $W_{\text{tot}}$, $T$ and $Z$, on the expected delay. When $W_{\text{tot}}$ nodes join $C$ channels randomly, $L_{W_{\text{tot}},C} = (W_{\text{tot}} + C - 1)!/[(C-1)!\,W_{\text{tot}}!]$ possible combinations can occur ($\forall i, 1 \le i \le L_{W_{\text{tot}},C}$: $[W_1(i) \dots W_C(i)]$).

To derive the possible combinations, we begin by assuming zero nodes in channels 1,2, … , $C-1$; this means that all $W_{\text{tot}}$ nodes must be in channel $C$. If zero nodes exist in channels 1,2, … , $C-2$ and one node exists in channel $C-1$, this means that $W_{\text{tot}} - 1$ nodes must be in channel $C$. Continuing on this expansion, we can cover all possible cases (two nodes in Ch$\{C-1\}$ and $W_{\text{tot}} - 2$ nodes in Ch$\{C\}$ and so on). This leads to the following summation for $C \ge 2$ and $W_{\text{tot}} \ge 2C$:

$$L_{W_{\text{tot}},C} = \sum_{i_1=0}^{W_{\text{tot}}}\sum_{i_2=0}^{W_{tot}-i_1}\dots\sum_{i_{C-2}=0}^{W_{\text{tot}}-\sum_{j=1}^{C-3} i_j}(W_{\text{tot}} - \sum_{j=1}^{C-2} i_j + 1) \tag{5.14}$$

Lemma 1: $L_{W_{\text{tot}},C} = \frac{(W_{\text{tot}}+C-1)!}{(C-1)!W_{\text{tot}}!}$.

*Proof of Lemma:* The proof is performed by induction for values of $C$. For $C = 2,3$, we validate straightforwardly that:

$C = 2$: $\frac{(W_{\text{tot}}+1)!}{W_{\text{tot}}!} = W_{\text{tot}} + 1$, and

$C = 3$: $\frac{(W_{\text{tot}}+2)!}{2W_{\text{tot}}!} = \frac{1}{2}(W_{\text{tot}} + 1)(W_{\text{tot}} + 2) = \sum_{i_1=0}^{W_{\text{tot}}}(W_{\text{tot}} - i_1 + 1)$.

Assuming that the lemma holds for $C = k$, i.e

$$\frac{(W_{\text{tot}}+k-1)!}{(k-1)!W_{\text{tot}}!} = \sum_{i_1=0}^{W_{\text{tot}}}\sum_{i_2=0}^{W_{\text{tot}}-i_1}\dots\sum_{i_{k-2}=0}^{W_{\text{tot}}-\sum_{j=1}^{k-3} i_j}(W_{\text{tot}} - \sum_{j=1}^{k-2} i_j + 1) \tag{5.15}$$

we shall show that we reach (5.15) with $k$ replaced by $k+1$.

We can write the case of $C = k+1$ from (5.14) as:

$$\sum_{i_1=0}^{W_{\text{tot}}}\left[\sum_{i_2=0}^{(W_{\text{tot}}-i_1)}\dots\sum_{i_{k-2}=0}^{(W_{\text{tot}}-i_1)-\sum_{j=2}^{k-3} i_j}\sum_{i_{k-1}=0}^{(W_{\text{tot}}-i_1)-\sum_{j=2}^{k-2} i_j}((W_{\text{tot}} - i_1) - \sum_{j=2}^{k-1} i_j + 1)\right]$$

$$= \sum_{i_1=0}^{W_{\text{tot}}} \frac{(W_{\text{tot}} - i_1 + k - 1)!}{(k-1)!\,(W_{\text{tot}} - i_1)!}$$

derived by using (5.15) for the sum series in brackets and assuming $(W_{\text{tot}} - i_1)$ total nodes. The last expression can be straightforwardly be rewritten as:

$$\sum_{i_1=0}^{W_{\text{tot}}} \frac{(W_{\text{tot}} - i_1 + k - 1)!}{(k-1)!\,(W_{\text{tot}} - i_1)!} = \frac{1}{(k-1)!} \sum_{i_1=0}^{W_{\text{tot}}} \frac{(i_1 + k - 1)!}{i_1!} = \frac{(W_{\text{tot}} + k)!}{(k-1)!\,k W_{\text{tot}}!}$$
$$= \frac{[W_{\text{tot}} + (k+1) - 1]!}{[(k+1) - 1]!\,W_{\text{tot}}!}$$

i.e. (5.15) with the replacement of $k$ by $k + 1$.

Thus, (5.15) holds for $k$ replaced by $k + 1$. As such, it holds for any $k \geq 2$. □

The probability of each combination $i$ occurring is $p(i)$, given by (5.11). Since nodes join a channel randomly, once each node makes a decision, it is a "success" or "fail" process for each channel: "success" if the node joins it, "fail" otherwise. The probability of "success" is $\frac{1}{C}$, while the probability of "fail" is $\frac{C-1}{C}$. Hence, for the first channel, the probability of having $W_1(i)$ nodes ("successes") out of $W_{\text{tot}}$ (based on the binomial distribution) is:

$$p_1(i) = \binom{W_{\text{tot}}}{W_1(i)} \frac{(C-1)^{W_{\text{tot}} - W_1(i)}}{C^{W_{\text{tot}}}}.$$

For the second channel, the probability of having $W_2(i)$ nodes out of $[W_{\text{tot}} - W_1(i)]$ possible nodes [since we assumed that $W_1(i)$ nodes have chosen to join the first channel] is:

$$p_2(i) = \binom{W_{\text{tot}} - W_1(i)}{W_2(i)} \frac{(C-1)^{W_{\text{tot}} - W_1(i) - W_2(i)}}{C^{W_{\text{tot}} - W_1(i)}}.$$

Iterating this for all channels, we derived in a similar fashion $p_3(i), \ldots, p_{C-1}(i)$. The remaining number of nodes, i.e. $[W_{\text{tot}} - \sum_{c=1}^{C-1} W_c(i)]$ nodes will be joining channel $C$ with probability $p_C(i) = 1$. Since these probabilities are independent, the probability of node distribution : $[W_1(i) \ldots W_C(i)]$ in channels $1, \ldots, C$ is:

$$p(i) = \prod_{c=1}^{C-1} p_c(i) = \prod_{c=1}^{C-1} \left[ \binom{W_{\text{res},c}(i)}{W_c(i)} \frac{(C-1)^{W_{\text{res},c}(i) - W_c(i)}}{C^{W_{\text{res},c}(i)}} \right]$$

with $W_{\text{res},c}(i) = W_{\text{tot}} - \sum_{m=1}^{c-1} W_m(i)$. Notice that the assumption of nodes deciding first on whether to join channel 1, then whether to join channel 2, etc., is not restrictive. In fact, the above analysis can be expressed with any order of channels without affecting the result.

Hence, the expected delay is $d_{W_{\text{tot}},C} = T\sum_{i=1}^{L_{W_{\text{tot}},C}} p(i) d_{\text{periods}}(i)$, with $d_{\text{periods}}(i)$ the expected number of periods until convergence to SS is achieved for combination $i$. For each combination, $d_{\text{periods}}(i)$ is dominated by the channel with the largest imbalance from the average, since this channel will have the largest inflow or outflow of nodes. The largest imbalance is expressed by $W_{\text{diff}}(i)$. The remainder of the proof estimates $d_{\text{periods}}(i)$. We present the case of the channel with the largest surplus of nodes under combination $i$ (assumed to be Ch$\{c\}$); the equivalent hold for the channel with the largest deficit.

First, nodes will desynchronize in Ch$\{c\}$, thus requiring $k_{\text{ss}}$ periods. Subsequently, they will gradually leave Ch$\{c\}$ until $[\![W_{\text{tot}}/C \pm 0.4\bar{9}]\!]$ nodes remain in that channel. Since nodes decide independently on whether to attempt a switch, the probability that of no switching within the first period is:

$$p_{\text{no\_sw},c}^{(0)} = \left(1 - p_{\text{sw},c}^{(0)}\right)^{W_{\text{diff}}(i) + [\![W_{\text{tot}}/C]\!]} \tag{5.16}$$

By construction, one switching attempt must happen within (maximally) $Z$ periods. Hence, the expected number of time periods until the first switch happens is:

$$d^{(1)} = \sum_{z=1}^{Z} z\left(p_{\text{no\_sw},c}^{(0)}\right)^{z-1}\left(1 - p_{\text{no\_sw},c}^{(0)}\right) + Z\left(p_{\text{no\_sw},c}^{(0)}\right)^{Z} \tag{5.17}$$

$$= \frac{1-\left(p_{\text{no\_sw},c}^{(0)}\right)^{Z}}{1-p_{\text{no\_sw},c}^{(0)}} \tag{5.18}$$

This is followed by two periods where nodes repeat their beacon message waiting for a "return" message. Iterating the above process, for the $k$th departure in Ch$\{c\}$, we reach $d^{(k)}$ given by (5.12). Finally, $d_{\text{periods}}(i)$ is found by the accumulation of all $W_{\text{diff}}(i)$ iterations, which leads to (5.10). □

## 5.2 Experiments

For our experiments, we used $W_{\text{tot}} = 16$ iMote2 sensors (with the 2.4GHz Chipcon CC2420 wireless transceiver), placed in an obstacle-free topology. All messages used the TinyOS standard with 96-byte payload. The utilized parameters were:

$b_{\text{thres}} = 0.02$, $T = 0.25$s, $\alpha = 0.95$, $\beta = 1.25$, $\forall c: p_{\text{sw},c}^{(0)} = 0.33$, $s_c^{(0)} = 1$, $Z = 60$. Due to the use of higher convergence threshold than the one used in DESYNC, we found $k_{\text{ss}} = 6$, which leads to significantly-faster convergence to SS than what is reported in [8]. All measurements are averages of several trials of 60s each. Up to $C = 8$ channels were used (out of the 16 available in IEEE802.15.4), and one base station is used per channel to passively record all messages for subsequent analysis.

Table 5.1 contains the results with respect to bandwidth efficiency (the last column of the table is discussed separately in the following paragraph). We also present the results of DESYNC [8], TSMP [32] (which is a centralized channel-hopping protocol) and the recently-proposed EM-MAC [44] in Table 5.2. These comprise the state-of-the-art in centralized and distributed channel hopping in WSNs. All approaches are realized over the same physical layer (IEEE802.15.4 and the Chipcon CC2420 transceiver). By comparing the two tables, it is evident that the total network throughput (throughput of all nodes) as well as the throughput per node is higher in the proposed TFDMA than in all the other TDMA or channel hopping solutions when all 8 channels are used. Our throughput surpasses DESYNC even in the single channel case because we use higher convergence threshold, leading to faster convergence to SS. Unlike EM-MAC that is designed for low-bandwidth wireless transmissions over lengthy periods of time, the proposed TFDMA can achieve very high bandwidth for rapid message exchanges within short intervals. This is very suitable for WSN-based surveillance and monitoring, where infrequent alerts can initiate rapid wake-up and high volume of WSN traffic for short intervals, before the network suspends again.

| **Total Channels** | **1** | **2** | **4** | **8** | **8, hidden node & reshuffling** |
|---|---|---|---|---|---|
| Tot. throughput (kbps) | 126.9 | 266.7 | 543.8 | 801.9 | 649.0 |
| Max per node (kbps) | 8.3 | 16.7 | 34.1 | 58.1 | 52.6 |
| Min per node (kbps) | 7.3 | 16.5 | 33.7 | 43.5 | 32.1 |
| Message loss (%) | 0.54 | 0.01 | 0.01 | 0.96 | 0.98 |

Table 5.1: Throughput of the proposed TFDMA with 16 nodes.

| Protocol | DESYNC [8] | TSMP [32] | EM-MAC [44] |
|---|---|---|---|
| Tot. throughput (kbps) | 55.0 | 574.4 | 5.1 |
| Max per node (kbps) | 3.5 | 35.9 (average) | 0.32 (average) |
| Min per node (kbps) | 3.2 | | |
| Message loss (%) | 0.30 | 0.01 | 0.00 |

Table 5.2: Throughput obtained with DESYNC, TSMP and EM-MAC; all results are reported under a fully-connected WSN topology comprising 16 nodes.

We also measured the average time to achieve convergence to SS in TFDMA versus the estimate of Proposition 2. Table 5.3 and Table 5.4 show the convergence time required by the other three solutions under comparison. Evidently, the proposed TFDMA achieves quick convergence, which agrees with the theoretical estimates of Proposition 2. Such low convergence times enable the application of node reshuffling (or suspension) in periodic intervals, i.e. all nodes can be forced to randomly join a new channel in order to increase their connectivity. By applying such node reshuffling every 60s, we obtained the results reported in the last column of Table 5.1; importantly, these results include the overhead of handling one-hop, possibly hidden, nodes based on the inclusion of neighboring nodes' beacon times within each node's beacon message, as proposed in [15]. These results still surpass the competing solutions despite the increase of beacon message size. A thorough study of properties of the proposed protocol under arbitrary topologies remains a topic for future work.

| Total Nodes | 16 | | | 8 | |
|---|---|---|---|---|---|
| Tot. Channels | 8 | 4 | 2 | 4 | 2 |
| Measured (s) | 4.7 [±1.7] | 4.0 [±1.0] | 3.2 [±0.5] | 3.1 [±0.7] | 2.9 [±0.6] |
| Proposition 2 (s) | 4.9 | 4.1 | 2.7 | 3.1 | 2.3 |

Table 5.3: Average delay (and standard error of mean) until SS.

| Protocol | DESYNC [8] | TSMP [32] | EM-MAC [44] |
|---|---|---|---|
| Delay until SS (s) | 8~48 | 48 | 8~9 |

Table 5.4: Average delay until SS under TSMP and EM-MAC.

```
1  get_RandNo = NODE_ADDRESS*call SysTime.getTime32()%10 + NODE_ADDRESS*NODE_ADDRESS
2  for (j=1; j<get_RandNo; j++){
3     rand_No = call Random.rand()
4  }
5  selected_channel = rand_No%number_channels + 1
```

Figure 5.3: Code for the random channel selection

## 5.3 Conclusion

We proposed a new distributed time-frequency division multiple access (TFDMA) protocol. By utilizing the concept of reactive listening, our approach distributes the available transmission opportunities in a balanced manner across time and frequencies (channels) in a sensor network without requiring the presence of a coordinator node. Stability and convergence time were derived analytically and validated experimentally based on TinyOS iMote2 wireless sensors. Our proposal allows for increased throughput and decreased convergence time versus TDMA-only schemes or versus centralized and distributed channel-hopping based approaches.

# Chapter 6
# Analytic Study of Energy Consumption in Desynchronization-based Wireless Sensor Networks under Variable Data Production Rates

In this chapter, we focus on the common application scenario of a monitoring infrastructure where sensor nodes follow a periodic duty cycle in order to capture and transmit measurements to a base station, or to another node that relays the information to a base station. We derive a parametric model for energy consumption in function of the system settings under the assumption of a uniformly-formed WSN, i.e. a network of identical sensor nodes that are: *(i)* producing data traffic with the same statistical characterization and *(ii)* directly connected to the base station represented by a symmetric star graph with balanced bandwidth allocation per link [47]. Within this framework, the key advance of our work in comparison to previous work on optimal energy management policies [73][74][75][78] is that we provide closed-form expressions for the minimum-required energy consumption of each sensor in a uniformly-formed WSN operating under a desynchronization-based collision-free communications protocol.

Energy consumption is the important issue beyond the topic of the convergence delay for this our thesis. In order to study our developments and validation of desynchronization-based protocols with energy-efficient operation, we proposed an analytic framework for characterizing practical energy consumption in uniformly-formed WSNs in this chapter.

## 6.1 Description of Systems under Consideration

Energy efficiency is a major challenge in WSNs. The approach pursued in this chapter proposes optimal energy management policies under given sensing and

transmission capabilities [70]-[76]. Our study is inspired by dynamic power management approaches [78]-[80]. The reader is referred to Section 1.2.4 that discussed related work to DPM and energy management systems.

For our analysis, we assume that, for an operational interval of $T$ seconds, the sensor nodes are continuously active for $T_{\text{act}}$ seconds. This defines the *duty cycle*

$$c = \frac{T_{\text{act}}}{T} \tag{6.1}$$

This activation can be triggered by external events or by scheduled data gathering with rate $c$ over the duration of the application, $0 < c < 1$. Examples are: data acquisition and transmission in environmental monitoring [62][98], event-driven activation for surveillance [70], and adaptive control of duty cycling for energy management [78][76]. Thus, the value of $c$ can be adjusted statically or dynamically based on empirical observations from the application environment.

When the sensor nodes are activated, they first converge into a *balanced time-frequency steady-state mode*, where each node joins one base station on a particular channel such that the number of nodes coupled to each base station is balanced and each base station can receive data from $n$ nodes without collisions. Several low-energy (centralized or distributed) WSN protocols, such as EM-MAC [44], wirelessHART [46], IEEE802.15.4 GTS [40]-[42][47] and TFDMA [34] can achieve this goal. For example, TFDMA, as presented in Chapter 5, achieves this for 16 nodes and 4 channels (i.e. $n = 4$) within 3-5 seconds [34], while the centralized IEEE802.15.4 GTS can establish collision-free single-channel time division multiple access (TDMA) within 1-2 seconds [47]. While energy is consumed for this convergence, the payoff for the WSN is the achievement of balanced, collision-free, steady-state operation with predictable characteristics during the active period. In this chapter, we focus on the basic case of a fully-connected network with one base station in order to facilitate the study of the energy consumption under a collision-free desynchronization-based communications protocol. An example of a uniformly-formed topology that can operate in collision-free steady-state mode are given in Figure 6.1.

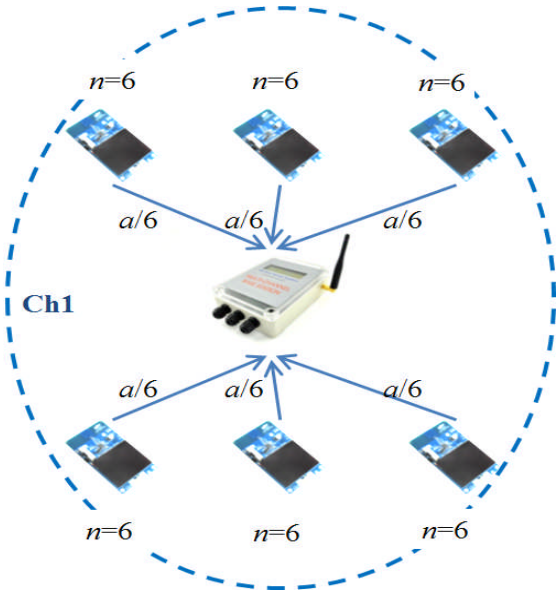


Figure 6.1: A uniformly-formed topology which is fully connected to one base staion with $a$ indicating the consumption rate of a base station (in bits-per-second)

Each sensor captures and transmits data with variable rate, which will be modelled as a random variable. The rate variability may stem from: adaptive sensing strategies [48], packet retransmissions or protocol adaptivity to mitigate interference effects [44], and variable-rate source-channel encoding [50], to reduce the transmission bit rate and ensure robustness to packet erasures [49]. Given that $n$ sensor nodes communicate with the same sink node without collisions during the steady-state, depending on the amount of data to be transmitted, a node may need to: *(i)* stay awake (beaconing and radio on) if less bits have to be sent than what is possible within its transmission slot; *(ii)* buffer the residual data if more bits must be sent than what its slot permits. Once the active period of $T_{\text{act}}$ seconds lapses, each node suspends its activity in order to preserve energy for $(1-c)T$ seconds. Figure 6.2 shows an example of the operation of the active period, which is discussed in detail in the next subsection.

We remark that practical WSN transceiver hardware reacts in intervals proportional to one packet transmission (or to the utilized time-frequency slotting mechanism). Thus, the transmission and reception of data is not strictly a continuous process.

However, energy consumption within each sensor node is strictly continuous as, regardless of the transceiver, each sensor node is active for the entire duration of $T_{\text{act}}$ seconds by sensing, processing data (e.g. to remove noise or to perform data encoding) and other runtime operations related to data gathering, processing and transmission (such as buffer management at the application, medium access and physical layers and servicing interrupts of the runtime environment).

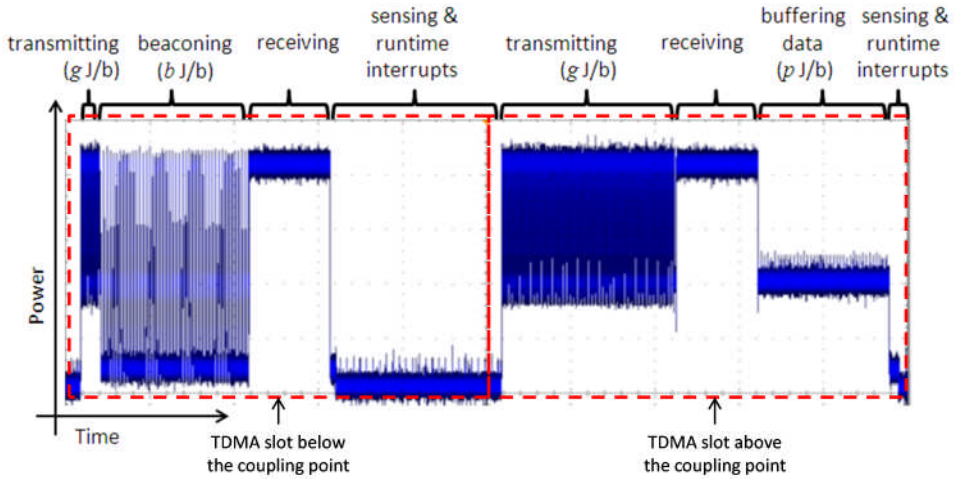


Figure 6.2: Energy profile of a TelosB sensor node running balanced TDMA data transmission for a fully-connected topology during the active period.

### 6.1.1 Data Consumption and Penalty

When the WSN goes into the active state, we assume that $k$ Joule is consumed by each sensor node in order to reach the balanced, collision-free, steady-state operation via one of the well-known centralized or distributed mechanisms suitable for this purpose [32],[34],[44],[46],[51], many of which were described in the previous chapters of this thesis. During the steady-state operation of each node, the average energy rate consumed to process and transmit data is $g$ Joule-per-bit.

Because the data production and transmission by each sensor node is a non-deterministic process, the data transmission rate (in bits-per-second) is modeled by random variable (RV) $\Psi$ with PDF $P(\psi)$. The statistical modeling of this rate can be gained by observing the occurred physical/chemical phenomena and analyzing the behavior of each node when it captures, processes and transmits bits relevant to them. Alternatively, the data transmission rate can be controlled (or "shaped")

by the system designer in order to achieve a certain goal, such as limiting the occurring latency or, in our case, to minimize the preserved energy required in order to operate each node in perpetuity.

| Symbol | Unit | Definition |
|---|---|---|
| $c$ | -- | Duty cycle |
| $T, T_{\text{act}}$ | s | Operational time interval, active time interval |
| $n$ | -- | Total number of sensor nodes communicating with a base station of the single-hop topology |
| $k$ | J | Energy consumed for wake-up, set-up and convergence |
| $g$ | J/b | Power for processing and transmitting one bit |
| $p$ | J/b | Penalty power for storing one bit during sink overloading |
| $b$ | J/b | Power during idle periods for the time interval corresponding to one bit transmission |
| $a$ | bps | Data consumption rate of a base station |
| $r$ | bps | Average data transmission rate per node |
| $\Psi \sim P(\psi)$ | bps | RV modeling the data production and transmission rate per node |
| $E[\psi]$ | bps | Expected data production and transmission rate per node |
| $P_{\text{battery}}$ | W | Required battery power supply for each node during the operational time interval |

Table 6.1: Nomenclature table

The data consumption rate of the application layer of a base station under the employed collision-free steady-state operation is $a$ bits-per-second (bps). For example, under the IEEE802.15.4 physical layer and the CC2420 transceiver, $a \cong 144$ kbps at the application layer under the NullMAC and NullRDC options of Contiki operating system[8]. Since each sink node is coupled with $n$ identical sensor nodes in the single-hop topology (Figure 6.1), we define the ratio $\frac{a}{n}$ as the coupling point of a base station node. This means that, in the ideal case, each sensor node should transmit its captured data at the rate of $\frac{a}{n}$ bps. However, given the time-varying nature of the data transmission rate per node, beyond the energy for data

[8]http://www.sics.se/contiki/wiki/index.php/Change_MAC_or_Radio_Duty_Cycling_Protocols contains more details; the NullMAC mechanism does not do any MAC-level processing and leads to the maximum energy efficiency, assuming that the application layer handles the transmission opportunities appropriately.

capturing and transmission we encounter the following two cases: *(i)* receiver underloading, where $\Psi < \frac{a}{n}$ and "idle" energy is consumed by the node with rate $b$ Joule-per-bit (J/b) by staying active during transmission opportunities for synchronization and other runtime purposes (e.g. transmitting beacon messages [34], [51]); *(ii)* receiver overloading, where $\Psi > \frac{a}{n}$ and "penalty" energy is consumed with rate $p$ J/b by the sensor to buffer (and retrieve) the data prior to transmission. Examples of both are illustrated in Figure 6.2 for TDMA-based collision-free transmission [8],[34]. The nomenclature summary of our system model is given in Table 6.1.

## 6.2 Characterization of Energy Consumption

We derive the analytic conditions that correspond to the minimum energy consumption required in the system model described previously. There are two modes of operation with complementary energy profiles: the active mode, where energy is consumed, and the sleep mode, where each node is suspended to save energy.

During the active mode period of $cT$ seconds we define four components for the energy consumption for each sensor node.

1. ***Setup and convergence energy*****:** Each node is activated once during the operational time interval. Thus the energy to converge to steady state is $k$ J. We remark that the convergence time is at least two orders of magnitude smaller than $T_{\text{act}}$ (e.g. 1-5 s vs. $T_{\text{act}} = 400$ s) and can be considered negligible in comparison to $T_{\text{act}}$.

2. ***Energy for sensing, processing and transmitting*** the node's own data, given by $cTg\int_0^{\infty} \psi P(\psi) d\psi = cTgE[\psi]$ J, with $E[\psi]$ the expected transmission rate of each node. If $E[\psi] > \frac{a}{n}$ (i.e. the mean transmission rate is higher than the coupling point), then $T_{\text{act}}$ includes the time each node has to remain active without producing new data, in order to complete the transmission of the data buffered in its flash memory.

3. ***Idle energy,*** consumed when the data rate $\Psi$ is smaller than the sink coupling point $\frac{a}{n}$ : $cTb\int_0^{\frac{a}{n}}(\frac{a}{n}-\psi)\,P(\psi)d\psi$ J. This energy corresponds to beaconing for synchronization and other runtime operations carried out during the transmit mode.

4. ***Penalty energy,*** consumed when the data rate $\Psi$ is larger than the sink coupling point $\frac{a}{n}$ and the data is buffered in high-power, typically off-chip, memory prior to transmission at the next available opportunity: $cTp\int_{\frac{a}{n}}^{\infty}(\psi-\frac{a}{n})\,P(\psi)d\psi$ J.

Notice that, apart from the setup and convergence energy, the energy consumption for all the number of nodes in WSNs is given in $E_c$. This energy consumption of each node is defined as the consumed energy which needs to have the produced or preserved energy to compensate for surviving in the system during the operational time interval. It can be calculated for each sensor nodes by:

$$E_c = k + cT\left[E[\psi]g + b\int_0^{\frac{a}{n}}\left(\frac{a}{n}-\psi\right)P(\psi)d\psi + p\int_{\frac{a}{n}}^{\infty}\left(\psi-\frac{a}{n}\right)P(\psi)d\psi\right] \tag{6.2}$$

Adding and subtracting $cTp\int_0^{\frac{a}{n}}\left(\psi-\frac{a}{n}\right)P(\psi)d\psi$ in $E_c$, we get

$$E_c = k + cT\left[E[\psi]g + b\int_0^{\frac{a}{n}}\left(\frac{a}{n}-\psi\right)P(\psi)d\psi - p\int_0^{\frac{a}{n}}\left(\psi-\frac{a}{n}\right)P(\psi)d\psi + p\int_0^{\frac{a}{n}}\left(\psi-\frac{a}{n}\right)P(\psi)d\psi + p\int_{\frac{a}{n}}^{\infty}\left(\psi-\frac{a}{n}\right)P(\psi)d\psi\right]$$

$$E_c = k + cT\left[E[\psi]g + b\int_0^{\frac{a}{n}}\left(\frac{a}{n}-\psi\right)P(\psi)d\psi + p\int_0^{\frac{a}{n}}\left(\frac{a}{n}-\psi\right)P(\psi)d\psi + E[\psi]p - \frac{ap}{n}\right]$$

Finally, we get

$$E_c = k + cT\left[E[\psi]\,(g+p) - \frac{ap}{n} + (b+p)\int_0^{\frac{a}{n}}\left(\frac{a}{n}-\psi\right)P(\psi)d\psi\right] \tag{6.3}$$

Evidently, the energy consumption depends on the coupling point, $\frac{a}{n}$, as well as on the PDF of the data transmission rate per sensor node, $P(\psi)$. In the remainder of this section, we consider different cases for $P(\psi)$ to derive the energy consumption under different statistical characterizations for the data transmission rate of each node.

We can now consider four PDFs (Uniform, Pareto, Exponential, Half-Gaussian) that have been used to model the marginal statistics of many real-world data

transmission applications. We provide the obtained analytic calculation and results for each distribution. This facilitates comparisons of the minimum energy consumption required under different characterizations for the data rate.

### *A. Uniform Distribution*

When no knowledge of the underlying statistics of the data generation process exists, one can assume that $P(\psi)$ is uniform over the interval $[0,2r]$:

$$P_{\mathrm{U}}(\psi) = \frac{1}{2r} \tag{6.4}$$

The expected value of $\Psi$ is $E_{\mathrm{U}}[\psi] = r$ bps. For $\frac{a}{n} < 2r$, by using (6.4) in (6.3), we obtain

$$E_{c,\mathrm{U}} = k + cT\left[r\,(g+p) - \frac{ap}{n} + \frac{a^2(b+p)}{4rn^2}\right] \tag{6.5}$$

For $b, p \neq 0$ The first derivative of $E_{c,\mathrm{U}}$ to $n$ is

$$\frac{dE_{c,\mathrm{U}}}{dn} = cT\left[\frac{ap}{n^2} - \frac{a^2(b+p)}{2rn^3}\right] \tag{6.6}$$

For $n \in [1,\infty)$, the number of nodes for which $\frac{dE_{c,\mathrm{U}}}{dn} = 0$ is[9]

$$n_{0,\mathrm{U}} = \frac{a(b+p)}{2pr} \tag{6.7}$$

The second derivative of $E_{c,\mathrm{U}}$ is

$$\frac{d^2E_{c,\mathrm{U}}}{dn^2} = cT\left[-\frac{2ap}{n^3} + \frac{3a^2(b+p)}{2rn^4}\right] \tag{6.8}$$

By evaluating $\frac{d^2E_{c,\mathrm{U}}}{dn^2}$ for $n_{0,\mathrm{U}}$ nodes, we obtain

$$\frac{d^2E_{c,\mathrm{U}}}{dn^2}\left(n_{0,\mathrm{U}}\right) = \frac{8Tcp^4r^3}{a^2\,(b+p)^3} \tag{6.9}$$

which is positive. This means that $n_{0,\mathrm{U}}$ is the number of nodes that achieves the minimum energy consumption for this case, which is

$$\min\{E_{c,\mathrm{U}}\} = k + cTr\left[g + \frac{pb}{b+p}\right] \tag{6.10}$$

[9] We remak that, when used in a practical setting, the optimal value for the number of nodes must be rounded to the nearest integer. However, for exposition simplicity we do not explicitly indicate this rounding in our notation.

The last equation demonstrates that the minimum power supply required over the operational interval is given by:

$$\min\{P_{\text{battery}}\}_{\text{U}} = \frac{k}{T} + cr\left[g + \frac{pb}{b+p}\right] \tag{6.11}$$

with $P_{\text{battery}}$ the expected power available to each node (in Watt). Hence, if the power obtained from the battery of each node is (at least) $\min\{P_{\text{battery}}\}_{\text{U}}$ W (averaged over the interval of $T$ seconds), this suffices for the operation of a WSN comprising $n_{0,\text{U}}$ nodes in the fully-connected network, with each node transmitting data with uniform rate between $[0,2r]$ bps. The minimum power shown in (6.11) is obtained under the operational parameters: $c, T, k, g, b, p$ (see Table 6.1) and $n_{0,\text{U}}$ nodes and $E_{\text{U}}[\psi] = r$ . These parameters can be derived based on the utilized technology and the application specifics, as we shall show in the next subsection.

*B. Pareto Distribution*

This distribution has been used, amongst others, to model the marginal data size distribution of TCP sessions that contain substantial number of small files and a few very large ones [52], [53]. Consider $P_{\text{P}}(\psi)$ as the Pareto distribution with scale $v$ and shape $\alpha \geq 2$ $(\alpha \in \text{N})$ ,

$$P_{\text{P}}(\psi) = \begin{cases} \alpha \dfrac{v^{\alpha}}{\psi^{\alpha+1}}, \psi \geq v \\ 0, \quad \text{otherwise} \end{cases} \tag{6.12}$$

The expected value of $\Psi$ is $E_{\text{P}}[\psi] = \frac{\alpha v}{\alpha - 1}$ bps. Thus if we set

$$v = \frac{\alpha - 1}{\alpha} r \tag{6.13}$$

we obtain $E_{\text{P}}[\psi] = r$ bps, i.e. we match the expected data transmission rate to that of the Uniform distribution. For the case of the Pareto distribution and $\frac{a}{n} \geq v$, we obtain via (6.3)

$$E_{c,\text{P}} = k + cT\left[\frac{\alpha v(g+p)}{\alpha - 1} + \frac{ab}{n} + (b+p)\left(\frac{v^{\alpha} n^{\alpha-1}}{a^{\alpha-1}(\alpha-1)} - \frac{\alpha v}{\alpha - 1}\right)\right] \tag{6.14}$$

The first derivative of $E_{c,\text{P}}$ to $n$ is

$$\frac{dE_{c,\mathrm{P}}}{dn} = cT\left[-\frac{ab}{n^2} + \frac{a}{n^2}(b+p)\left(\frac{nv}{a}\right)^{\alpha}\right] \tag{6.15}$$

The number of nodes for which $\frac{dE_{c,\mathrm{P}}}{dn} = 0$ is

$$n_{0,\mathrm{P}} = \left(\frac{a}{v}\right)\left(\frac{b}{b+p}\right)^{\frac{1}{\alpha}} \tag{6.16}$$

The second derivative of $E_{c,\mathrm{P}}$ is

$$\frac{d^2E_{c,\mathrm{P}}}{dn^2} = cT\left[\frac{2ab}{n^3} + \frac{a}{n^3}(\alpha-2)(b+p)\left(\frac{nv}{a}\right)^{\alpha}\right] \tag{6.17}$$

By evaluating $\frac{d^2E_{c,\mathrm{P}}}{dn^2}$ for $n_{0,\mathrm{P}}$ nodes, we obtain

$$\frac{d^2E_{c,\mathrm{P}}}{dn^2}\left(n_{0,\mathrm{P}}\right) = \frac{cT(2b + (\alpha-2)(b+p)^{\frac{3}{\alpha}}v^3)}{a^2\, b^{\frac{3}{\alpha}}} \tag{6.18}$$

which is positive. This means that $n_{0,\mathrm{P}}$ is the number of nodes that achieves the minimum energy consumption for this case, which is

$$\min\{E_{c,\mathrm{P}}\} = k + cTr\left[g - b + b^{\frac{\alpha-1}{\alpha}}(b+p)^{\frac{1}{\alpha}}\right] \tag{6.19}$$

The last equation demonstrates that the minimum power supply required over the operational interval is given by:

$$\min\{P_{\text{battery}}\}_{\mathrm{P}} = \frac{k}{T} + cr\left[g - b + b^{\frac{\alpha-1}{\alpha}}(b+p)^{\frac{1}{\alpha}}\right] \tag{6.20}$$

A special case for this distribution is when $\alpha = r$, which leads to $v = (r-1)$ from (6.13). Then, the expected value of $\Psi$ is $E_{\mathrm{F}}[\psi] = r$ bps and its standard deviation is $\sigma_{\mathrm{F}}[\psi] = \sqrt{\frac{r}{r-2}}$. For $r > 150$ bps, the standard deviation is less than 0.7% of the mean value. Thus, in practice this case corresponds to transmission with fixed rate of $r$ bps. This scenario occurs in WSNs capturing and transmitting data with fixed rate during their active time, e.g. in periodic temperature or humidity measurements gathered by WSNs [99],[100]. For this case, the number of nodes leading to the minimum required power is:

$$n_{0,\mathrm{F}} = \left(\frac{a}{r-1}\right)\left(\frac{b}{b+p}\right)^{\frac{1}{r}} \tag{6.21}$$

For the vast majority of values for $a$ and $r$ used in practical WSN applications, $n_{0,\mathrm{F}}$ is equal to $\left\lfloor \frac{a}{r} + 0.5 \right\rfloor$ when rounded to the nearest integer. This agrees with the

intuitive answer for balancing fixed-rate transmission with $r$ bps to consumption rate of $a$ bps. This means that $n_{0,\mathrm{F}}$ is the number of nodes that achieves the minimum energy consumption for this case, which is

$$\min\{E_{c,\mathrm{F}}\} = k + cTr\left[g - b + b^{\frac{r-1}{r}}(b+p)^{\frac{1}{r}}\right] \tag{6.22}$$

*C. Exponential Distribution*

The marginal statistics of MPEG video traffic have often been modelled as exponentially decaying [54]. Consider $P_{\mathrm{E}}(\psi)$ as the Exponential distribution with rate parameter $\frac{1}{r}$

$$P_{\mathrm{E}}(\psi) = \frac{1}{r}\exp\left(-\frac{1}{r}\psi\right) \tag{6.23}$$

for $\psi \geq 0$ . In this case, the expected value of $\Psi$ is $E_{\mathrm{E}}[\psi] = r$ bps via (6.3), we obtain

$$E_{c,\mathrm{E}} = k + cT\left[r(g+p) + \frac{ab}{n} + r(b+p)(\exp\left(-\frac{a}{nr}\right) - 1)\right] \tag{6.24}$$

The first derivative of $E_{c,\mathrm{E}}$ to $n$ is

$$\frac{dE_{c,\mathrm{E}}}{dn} = cT\left[-\frac{ab}{n^2} + \frac{a}{n^2}(b+p)\exp\left(-\frac{a}{nr}\right)\right] \tag{6.25}$$

Assuming that $b \neq 0$ , the number of nodes for which $\frac{dE_{c,\mathrm{E}}}{dn} = 0$ is

$$n_{0,\mathrm{E}} = \frac{a}{r(\ln(\frac{b+p}{b}))} \tag{6.26}$$

The second derivative of $E_{c,\mathrm{P}}$ is

$$\frac{d^2E_{c,\mathrm{E}}}{dn^2} = cT\left[\frac{2ab}{n^3} + \frac{a}{n^4 r}(b+p)\exp\left(-\frac{a}{nr}\right)(a-2nr)\right] \tag{6.27}$$

By evaluating $\frac{d^2E_{c,\mathrm{E}}}{dn^2}$ for $n_{0,\mathrm{E}}$ nodes, we obtain

$$\frac{d^2E_{c,\mathrm{E}}}{dn^2}(n_{0,\mathrm{E}}) = \frac{cTbr^3}{a^2}\ln(\frac{b+p}{b})^4 \tag{6.28}$$

which is positive and the natural logarithm is raised to an even power. This means that $n_{0,\mathrm{E}}$ is the number of nodes that achieves the minimum energy consumption for this case, which is

$$\min\{E_{c,\mathrm{E}}\} = k + cTr\left[g + b(\ln\left(\frac{b+p}{b}\right))\right] \tag{6.29}$$

The last equation demonstrates that the minimum power supply required over the operational time interval for this case is given by:

$$\min\{P_{\text{battery}}\}_{\mathrm{E}} = \frac{k}{T} + cr\left[g + b(\ln\left(\frac{b+p}{b}\right))\right] \tag{6.30}$$

*D. Half-Gaussian Distribution*

We conclude this part by considering $P_{\mathrm{H}}(\psi)$ as the Half-Gaussian distribution with mean $E_{\mathrm{H}}[\psi] = r$ .

$$P_{\mathrm{H}}(\psi) = \begin{cases} 0, & \psi < 0 \\ \dfrac{2}{\pi r}\exp(\dfrac{\psi^2}{\pi r^2}), & \psi \geq 0 \end{cases} \tag{6.31}$$

This distribution has been used widely in data gathering problems in science and engineering when the modelled data has non-negativity constraints. Some recent examples include the statistical characterization of motion vector data rates in Wyner-Ziv video coding algorithms suitable for WSNs [55], or the statistical characterization of samples captured by an image sensor [56],[57]. Via (6.3), we obtain

$$E_{c,\mathrm{H}} = k + cT\left[r(g+p) - \frac{ap}{n} + (b+p)\left(r(\exp\left(-\frac{a^2}{\pi r^2 n^2}\right) - 1\right) + \frac{a}{n}\mathrm{erf}(\frac{a}{\sqrt{\pi}nr})\right)\right] \tag{6.32}$$

The first derivative of $E_{c,\mathrm{H}}$ to $n$ is

$$\frac{dE_{c,\mathrm{H}}}{dn} = cT\left[\frac{ap}{n^2} - \frac{a}{n^2}(b+p)\mathrm{erf}\left(\frac{a}{\sqrt{\pi}nr}\right)\right] \tag{6.33}$$

with $\mathrm{erf}(\cdot)$ the error function that can be approximated by its Taylor series expansion. Under $b \neq 0$ and $p \neq 0$, the number of nodes that leads to the minimum power required under data transmission rate (per node) characterized by $P_{\mathrm{H}}(\psi)$

$$n_{0,\mathrm{H}} = \frac{a}{\sqrt{\pi}r\ \mathrm{erf}^{-1}(\frac{p}{b+p})} \tag{6.34}$$

with $\mathrm{erf}^{-1}(\cdot)$ the inverse error function, which can be approximated by its series expansion. Then the second derivative of $E_{c,\mathrm{H}}$ is

$$\frac{d^2E_{c,\mathrm{H}}}{dn^2} = cT\left[-\frac{2ap}{n^3} + \frac{2a}{n^3}(b+p)\mathrm{erf}\left(\frac{a}{\sqrt{\pi}nr}\right) + \frac{2a^2}{\pi rn^4}(b+p)\exp\left(-\frac{a^2}{\pi n^2 r^2}\right)\right] \quad (6.35)$$

By evaluating $\frac{d^2E_{c,\mathrm{H}}}{dn^2}$ for $n_{0,\mathrm{H}}$ nodes, we obtain

$$\frac{d^2E_{c,\mathrm{H}}}{dn^2}(n_{0,\mathrm{H}}) = \frac{2\pi cTr^3}{a^2}(b+p)\exp\left(-\left[\mathrm{erf}^{-1}\left(\frac{p}{b+p}\right)\right]^2\right)\left[\mathrm{erf}^{-1}\left(\frac{p}{b+p}\right)\right]^4 \quad (6.36)$$

which is positive since the inverse error function is raised to an even power and all variables are positive. This means that $n_{0,\mathrm{H}}$ is the number of nodes that achieves the minimum energy consumption for this case, which is

$$\min\{E_{c,\mathrm{H}}\} = k + cTr\left[g - b + (b+p)\ \exp\left(-\mathrm{erf}^{-1}\left(\frac{p}{b+p}\right)^2\right)\right] \quad (6.37)$$

The last equation demonstrates that the minimum power supply required over the operational time interval for this case is given by:

$$\min\{P_{\mathrm{battery}}\}_{\mathrm{H}} = \frac{k}{T} + cr\left[g - b + (b+p)\ \exp\left(-\mathrm{erf}^{-1}\left(\frac{p}{b+p}\right)^2\right)\right] \quad (6.38)$$

## 6.3 Evaluation of the Analytic Results

We consider a typical WSN setup comprising of several TelosB nodes (using the IEEE802.15.4 standard with the CC2420 transceiver) running the low-power Contiki 2.6 operating system. All nodes use our recently-proposed TFDMA protocol (which is available as open source [34],[45]) to communicate with the base station existing at the same channel, following topologies such as the ones shown in Figure 6.1. TFDMA can be deployed at the application layer with very low complexity and provides for balanced multichannel coordination of multiple nodes. We opted for its usage as it allows for quick convergence to the steady state and permits for collision-free communications once steady state has been established. It also provides for comparable or superior bandwidth utilization to channel-hopping approaches like TSMP and EM-MAC [34]. However, similar results can be obtained with any other protocol ensuring collision-free single- or multi-channel communications under a multi-level cluster hierarchy, such as TSMP [32], IEEE802.15.4 GTS [51],[58], etc.

For the utilized TFDMA and active time $T_{\text{act}} = 400$ s, convergence has been shown to occur in less than 1.3% of $T_{\text{act}}$ (3-5s) and, on average, the energy dissipation for convergence has been found to be $k = 165.6$ mJ in our setup. Concerning the communications side, following the default TFDMA setup, for all our measurements we set the packet size to 114 bytes, the DESYNC interval to 1s and the DESYNC constant to 0.65 [8]. Each node transmits 1-byte beacon packets every 8ms when it is not transmitting data packets during its transmission slot to maintain connectivity and synchronization. Finally, since the TFDMA protocol ensures no collisions occur during the steady-state active mode, we are utilizing the very-low complexity NullMAC and NullRDC options of Contiki RTOS, which lead to maximum data consumption rate at the application layer of $a = 144$ kbps.

Concerning the data gathering itself, we created artificial data via a custom Matlab function that, starting from the rand() function, generates data with Uniform, Pareto, Exponential and Half-Gaussian distributions (considered in the previous subsection) via rejection sampling [59], with mean transmission rate equal to $r = 24$ kbps. The data is copied onto each node and it is read from its external flash memory during the steady-state active mode. This ensures that: *(i)* we match the different PDFs under consideration and *(ii)* the energy to retrieve this data from the flash memory replaces the sensing energy (that would have been dissipated if the data had come from an actual sensing process).

Under these operational settings, our energy measurement setup comprises a high-tolerance 1 Ohm resistor placed in series with each TelosB node. By measuring the current consumption at the resistor and knowing that each node operates at 3 Volt, we derive the real-time energy consumption (see Figure 6.2 for examples). The utilized time resolution for the power measurements was 10 KHz using a Tektronix MDO4104-6 oscilloscope. Under this setup, we also measured the different energy rates of Table 6.1 by enabling transmission, listening, writing to flash memory and beaconing during the idle state to maintain synchronization.

They were found to be: $g = 2.29262 \times 10^{-7}$ J/b , $p = 3.89392 \times 10^{-7}$ J/b and $b = 2.17324 \times 10^{-7}$J/b .

The results of Figure 6.3 demonstrate that each transmission rate distribution incurs different energy consumption and the ranking of the data production and transmission PDFs in this respect is precisely:

**Fixed Rate < Pareto < Uniform < Half − Gaussian < Exponential**

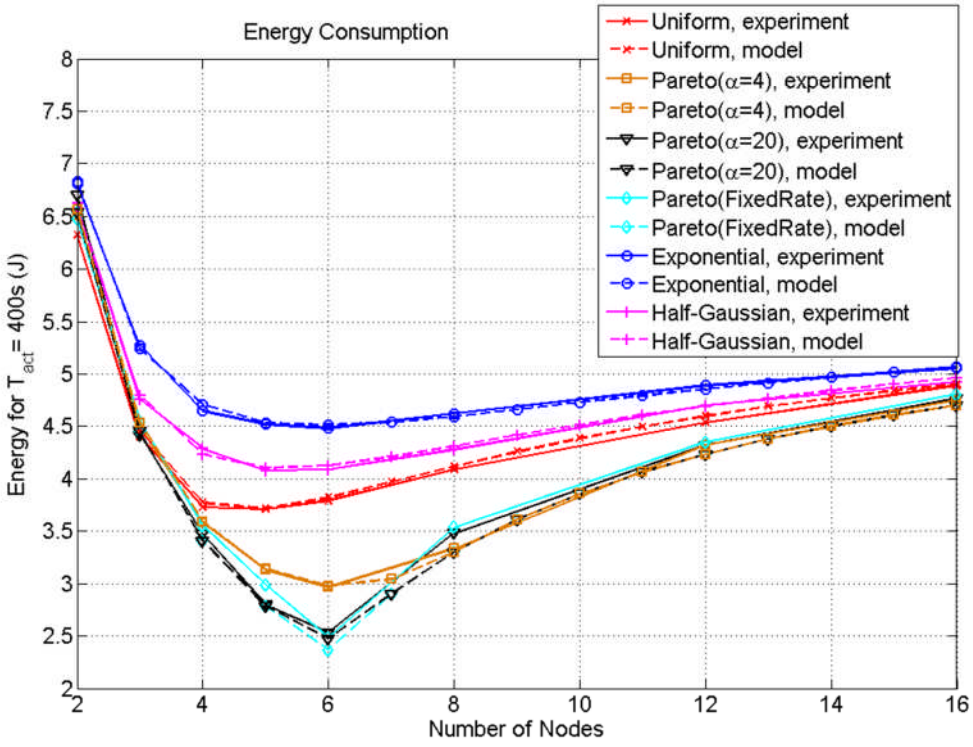


Figure 6.3: Energy consumption per node with different data transmission rates and under different numbers of nodes

| minimum energy consumption of difference transmission rate | Theoretical (J) | Experimental (J) | Percentile error (%) |
|---|---|---|---|
| Uniform, $n_{0,U} = 4.67$ | 3.705 | 3.707 | 0.042 |
| Pareto ($\alpha = 4$), $n_{0,P} = 6.19$ | 2.977 | 2.965 | 0.391 |
| Pareto ($\alpha = 20$), $n_{0,P} = 6.00$ | 2.476 | 2.531 | 2.206 |
| Fixed Rate, $n_{0,F} = 6.00$ | 2.367 | 2.484 | 4.951 |
| Exponential, $n_{0,E} = 5.84$ | 4.508 | 4.482 | 0.579 |
| Half-Gaussian, $n_{0,H} = 5.21$ | 4.099 | 4.079 | 0.494 |

Table 6.2: The minimum energy consumption required amongst the considered PDFs for activation time $T_{\text{act}} = 400$ s (6 min 40 s)

Thus, the manner the data traffic is shaped in a WSN plays an important role in the system's requirements for minimum energy consumption. Moreover, the results show that, depending on the transmission rate PDF, the number of nodes where the minimum energy consumption occurs, i.e. $n_{0,\mathrm{U}}, n_{0,\mathrm{P}}, n_{0,\mathrm{F}}, n_{0,\mathrm{E}}$ and $n_{0,\mathrm{H}}$ (as seen in (6.7), (6.16), (6.21), (6.26) and (6.34) respectively), may differ. The accuracy of these analytic estimations is quantified in Table 6.2 in comparison to the experimentally-obtained values for the minimum energy consumption of each distribution with the theoretical solution in (6.11), (6.20), (6.22), (6.30) and (6.38).

## 6.4 Conclusion

We proposed an analytic framework for characterizing practical energy consumption in uniformly-formed WSNs. Our framework recognizes the importance of the application data transmission rate in the WSN's energy dissipation. This framework provides for an analytic assessment of the expected energy dissipation in function of the system parameters, under a variety of statistical characterizations for the data transmission rate of each sensor node. The experimental assessment via low-power TelosB node validates that our analytical framework matches experiments with accuracy that within 7% of the energy consumption. This can be used in conjunction with future power management systems in WSN nodes in order to predict the best possible data frame rate and transmission rate that can be accommodated in function of the system's operational settings.

# Chapter 7
# Study of Energy Consumption for Distributed Coordination in Visual Sensor Networks

The integration of low-power wireless networking technologies such as WSNs with inexpensive complementary metal-oxide semiconductors (CMOS) cameras has enabled the development of the so called Visual Sensor Networks (VSNs), that is, networks of wireless devices capable of sensing multimedia content such as still images and video, as well as scalar sensor data from the environment. Due to their flexibility and low-cost, VSNs may potentially enable new applications ranging from enhanced surveillance to advanced service for health care, thus they have attracted the interest of researchers worldwide in the last few years.

VSNs are uniquely challenging with respect to traditional WSNs, because of the struggle between the application requirements and hardware constraints. Multimedia applications require to process, store and transmit large amount of data, which can be extremely demanding for resource-limited VSN hardware. Hence, energy consumption plays a crucial role in the design of a VSN, especially for those applications where a VSN is required to operate for hours of even days perpetually. Again, in the last few years several works have addressed the problem of lifetime maximization in VSNs: depending on the research area, solutions are available for energy-aware protocols [127], cross-layer optimization [128][129], application trade-offs [130] and deployment strategies [131].

In this chapter, we approach the problem of energy minimization in a more system-oriented manner. We focus on homogeneous VSNs composed by identical sensors and we derive a parameterized analytic model that captures the expected energy consumption in function of *(i)* the number of visual nodes deployed, *(ii)* the frame-rate used on board of each camera and *(iii)* the statistical characterization of

the multimedia traffic. We also take into account application-specific constraints, such as spatial/temporal resolution or budget constraints. The resulting energy equations can be solved by means of minima analysis, to provide closed-form expressions for the minimum energy consumption.

The framework in this chapter has been developed from previous chapter and also changed some parameters which are $k$ (the number of frames by each camera) and $a$ (Energy for data acquisition one bit) and added a new parameter that is $s_{\mathrm{b}}$ (data consumption rate of a base station). The rest of this chapter is organized as follows: in Section 7.1 we present the analytic model that characterizes the system under consideration. In Section 7.2, we derive closed-form expression for minimum energy consumption under different statistical characterization for the multimedia traffic produced by the camera sensors, and under application-drive constraints. Then, in Section 7.3 we validate the proposed model for the radio subsystem through experimental measurements on a real sensor network testbed via TelosB node. Two different real-case scenarios are presented in Section 7.4, where we show the benefits of using the proposed model in the design of a VSN. Finally, Section 7.5 concludes the chapter.

## 7.1 System Model

We consider a wireless visual sensor network organized in a star topology, with $n$ camera nodes sending multimedia data to a base station (sink) which is the center of the star. The network is operated by some form of collisions-free protocol (e.g. TDMA) and we assume that each camera node comprises two coupled subsystems, the multimedia subsystem and the radio subsystem. The multimedia subsystem is responsible for acquiring images, processing them and communicating the processed data to the radio subsystem, which transmits the multimedia data to the base station. Example multimedia applications that follow this scheme are: JPEG compression and transmission of still images [132], or visual features extraction, compression and transmission for object recognition [133]. The two subsystems work in parallel, that is, while the multimedia system acquires and processes data

for the time slot $t$, the radio subsystem transmits the multimedia data relative to the time slot $t-1$.

We analyze the energy consumption of such a system, and derive the optimal settings to minimize it, under two application constraints, namely:

• *spatial coverage bounds*: the number of deployed nodes communicating with a base station of the single-hop topology, $n$, is upper-bounded and lower-bounded, i.e. $n_{\min} \leq n \leq n_{\max}$

• *temporal coverage bounds*: the minimum frame acquisition rate, $f_{\min}$, within the time slot, $T$, is lower-bounded, i.e. $f_{\min} T \leq k$

In general, the data produces by each sensor node in such multimedia applications is a non-deterministic process. Thus we can model the amount of data produced per frame by each camera node with a random variable (RV) $\Psi$, with PDF $P(\psi)$.

Regardless to the specific application and the duty cycle between operational and active time interval, we can assume that each sensor node performs the following operations:

1. ***Acquisition***: a new frame is acquired by means of a low-power camera sensor. Each frame acquisition costs $a$ Joules. The energy consumed during the time slot, $T$, is hence $ka$ Joules.

2. ***Processing and transmission***: the frame is processed with a CPU-intensive algorithm by the multimedia subsystem. The results are communicated to the radio subsystem, which sends them to the base station. Let $g$ be the energy cost of processing and transmitting one bit of information (different applications may have different processing costs, while the transmission cost depends only on the specific radio chip used by the sensor node). The energy consumed is hence $g \int_0^{\infty} \psi_k \, P(\psi_k) d\psi_k = gE[\psi_k]$ Joules, with $P(\psi_k)$ is the RV that comprises the data generated by the processing and transmitting for the sum of $k$ frame and $E[\psi_k]$ the expected data generated by the processing and transmitting per node for the sum of $k$ frame.

3. ***Buffering or Idling***: The sensor network consists of $n$ sensor nodes organized with a collision-free protocol. When $s_{\mathrm{b}}$ is the data consumption rate (capacity) of the base station in bits per second. During time slot, $T$, each sensor node can transmit $\frac{s_{\mathrm{b}}T}{n} = \frac{s}{n}$ bits successfully, with $s$ the data consumed in bits by a base station within time slot. Thus we identify two cases: if the amount of data generated by the processing phase is greater than $\frac{s}{n}$ the sensor node has to buffer the remaining data in a high-power, typically off-chip, memory. Let $p$ be the energy cost of storing one bit of the data generated, the energy spent for buffering is $p\int_{\frac{s}{n}}^{\infty}(\psi_k - \frac{s}{n})\,P(\psi_k)d\psi_k$ Joules. Conversely, if the data generated is less than $\frac{s}{n}$, the sensor node can enter in an idle state where an energy $b$ is consumed for beaconing and other synchronization operations. Hence, the consumed energy is $b\int_0^{\frac{s}{n}}(\frac{s}{n} - \psi_k)\,P(\psi_k)d\psi_k$ Joules. Note that here we have used a random variable $P(\psi_k)$ to model the data generated by $k$ frame acquisitions.

Summing all the contributions, the energy consumption of the coupled system $E_{\mathrm{c}}$ is:

$$E_{\mathrm{c}} = ka + gE[\psi_k] + p\int_{\frac{s}{n}}^{\infty}(\psi_k - \frac{s}{n})\,P(\psi_k)d\psi_k + b\int_0^{\frac{s}{n}}(\frac{s}{n} - \psi_k)\,P(\psi_k)d\psi_k \tag{7.1}$$

Adding and subtracting $p\int_0^{\frac{s}{n}}(\psi_k - \frac{s}{n})\,P(\psi_k)d\psi_k$ to (7.1) leads to:

$$E_{\mathrm{c}} = ka + (g+p)E[\psi_k] - \frac{ps}{n} + (b+p)\int_0^{\frac{s}{n}}(\frac{s}{n} - \psi_k)\,P(\psi_k)d\psi_k \tag{7.2}$$

## 7.2 Derivation of the Minimum Energy Consumption under Application Constraints

In the following, we will consider different cases for $P(\psi_k)$ and derive the best choice for $n$ and $k$ that allows to minimize the energy consumption, while ensuring the conditions imposed by the spatial and temporal coverage constraints. The objective is to minimize $E_{\mathrm{c}}$ subject to the spatial and temporal constraints defined in previous section, that is:

$$\text{Minimize } E_{\mathrm{c}}(n,k)$$

subject to:

$$n_{\min} \le n \le n_{\max}$$

$$f_{\min} T \leq k$$

### 7.2.1 Illustrative Case: Uniform Distribution

In the simplest case, one can assume that $P_{\mathrm{U}}(\psi_k)$ is uniform over the interval $[0, 2kr]$, being $r$ the expected value of $\Psi$, in bits. We can write:

$$P_{\mathrm{U}}(\psi_k) = \begin{cases} \frac{1}{2kr}, 0 \leq \psi_k \leq 2kr \\ 0, \quad \text{otherwise} \end{cases} \tag{7.3}$$

Using (7.3) in (7.2) with $E_{\mathrm{U}}[\psi_k] = kr$ ,we obtain:

$$E_{\mathrm{c,U}} = k[a + (g + p)r] - \frac{ps}{n} + \frac{s^2(b + p)}{4n^2 kr} \tag{7.4}$$

We start by searching for critical points of $E_{\mathrm{c,U}}$, which are the candidates for being minima. By definition, a critical point of a multidimensional function is the point where the gradient of the function itself is equal to zero. Hence, we have to impose that both the derivatives of $E_{\mathrm{c,U}}$ with respect to $n$ and $k$ are zero.

$$\begin{cases} \frac{\partial E_{\mathrm{c,U}}}{\partial n} = \frac{ps}{n} - \frac{s^2(b + p)}{4n^3 kr} = 0 \\ \frac{\partial E_{\mathrm{c,U}}}{\partial k} = [a + (g + p)r] - \frac{s^2(b + p)}{4n^2 k^2 r} = 0 \end{cases} \tag{7.5}$$

The only solution for the system equation in (7.5) turns out to be $a$ negative, which is not feasible since $a$ is an energy cost. Hence we conclude that there is no solution $S \in \mathbb{R}\mathrm{x}\mathbb{R}$ with $S \equiv (n, k)$ such that the gradient is zero. In other words, none of the point inside the domain of $E_{\mathrm{c,U}}$ is a minimum. Hence, what we do is to look at one or the other direction individually (i.e., $n$ or $k$) with the general idea to find a minimum at least for that particular direction and then picking up , for the other direction, the best value that doesn't violate the constraints.

#### 7.2.1.1. $n$ Direction

Let's cut the function $E_{\mathrm{c,U}}$ with a plane $k = \bar{k}$, and evaluate the cut $E_{\mathrm{c,U}}(n, \bar{k})$ which is now function of $n$ only. The minimum of $E_{\mathrm{c,U}}(n, \bar{k})$ is

$$n_{0,\mathrm{U}} = \frac{s(b + p)}{2p\bar{k}r} \tag{7.6}$$

To generalize to any $\bar{k}$, let's evaluate $E_{\mathrm{c,U}}(n, k)$ on $n_{0,\mathrm{U}}$:

$$E_{\mathrm{c,U}}(n_{0,\mathrm{U}}, k) = k\left[a + (g + p - \frac{p^2}{b+p})r\right] \tag{7.7}$$

which has its minimum value for the minimum allowable $k$. Since the constraint on the minimum frame rate requires that $k \geq f_{\min}T$, the solution turns out to be

$$S_{n_{0,\mathrm{U}}} = \left(\frac{s_{\mathrm{b}}(b+p)}{2pf_{\min}r}, f_{\min}T\right) \tag{7.8}$$

is the optimal value in the $n$ direction.

### 7.2.1.2. $k$ Direction

Similarly, let's cut the function $E_{\mathrm{c,U}}$ with a plane $n = \bar{n}$, and minimize $E_{\mathrm{c,U}}(\bar{n}, k)$ which is now function of $k$ only. The minimum analysis gives

$$k_{0,\mathrm{U}} = \frac{s\alpha}{2\bar{n}} \tag{7.9}$$

As a minimum with $\alpha = \sqrt{\frac{b+p}{r[a+(g+p)r]}}$. Again, to generalize to any $\bar{n}$ direction, let's evaluate $E_{\mathrm{c,U}}(n, k)$ on $k_{0,\mathrm{U}}$:

$$E_{\mathrm{c,U}}(n, k_{0,\mathrm{U}}) = \frac{s}{n}\sqrt{\frac{(b+p)(a+(g+p)r)}{r}} - p \tag{7.10}$$

which has its minimum value for the maximum allowable $n$, which in our case is $n_{\max}$. Hence,

$$S_{k_{0,\mathrm{U}}} = \left(n_{\max}, \frac{s\alpha}{2n_{\max}}\right) \tag{7.11}$$

is the optimal value in the $k$ direction.

### 7.2.1.3. Uniqueness of Solution

So far, we have found two potential solutions for our problem: $S_{n_{0,\mathrm{U}}}$ which minimize the energy when looking at the $n$ direction only, and $S_{k_{0,\mathrm{U}}}$ which does the same for the $k$ direction. However, it is not clear which is the best solution. It turns out that using $S_{n_{0,\mathrm{U}}}$ or $S_{k_{0,\mathrm{U}}}$ depends only on the particular choice of $f_{\min}$.

Recalling that any solution $(n, k)$ must respect *(i)* $n \geq n_{\min}$, *(ii)* $n \leq n_{\max}$ and *(iii)* $k \geq f_{\min}T$, we consider $S_{n_{0,\mathrm{U}}}$, imposing the constraint:

$$n_{\min} \leq \frac{s_{\mathrm{b}}(b+p)}{2pf_{\min}r} \leq n_{\max} \tag{7.12}$$

which leads to

$$\frac{s_{\mathrm{b}}(b+p)}{2prn_{\max}} \leq f_{\min} \leq \frac{s_{\mathrm{b}}(b+p)}{2prn_{\min}} \tag{7.13}$$

or, by using two new variables $f_{1,\mathrm{U}}$ and $f_{2,\mathrm{U}}$

$$f_{1,\mathrm{U}} \leq f_{\min} \leq f_{2,\mathrm{U}} \tag{7.14}$$

with $f_{1,\mathrm{U}}$ is clearly smaller than $f_{2,\mathrm{U}}$. Similarly, by looking at $S_{k_{0,\mathrm{U}}}$, we obtain:

$$\frac{s\alpha}{2n_{\max}} \geq f_{\min}T \tag{7.15}$$

which leads to

$$f_{\min} \leq \frac{s_{\mathrm{b}}\alpha}{2n_{\max}} \tag{7.16}$$

with using a new variable $f_{0,\mathrm{U}}$. We proved that $f_{0,\mathrm{U}} < f_{1,\mathrm{U}}$.

Hence the optimal solutions are $S_{n_{0,\mathrm{U}}}$ for (7.14) and $S_{k_{0,\mathrm{U}}}$ for (7.16). Now we consider the rest of the cases when $f_{0,\mathrm{U}} < f_{\min} < f_{1,\mathrm{U}}$ and when $f_{\min} > f_{2,\mathrm{U}}$. In both cases, neither $S_{n_{0,\mathrm{U}}}$ nor $S_{k_{0,\mathrm{U}}}$ can be used: because $n_{0,\mathrm{U}} > n_{\max}$ when using $S_{n_{0,\mathrm{U}}}$ and $k_{0,\mathrm{U}} < f_{\min}T$ when using $S_{k_{0,\mathrm{U}}}$ for the first case and because $n_{0,\mathrm{U}} < n_{\min}$ when using $S_{n_{0,\mathrm{U}}}$ and $k_{0,\mathrm{U}} < f_{\min}T$ when using $S_{k_{0,\mathrm{U}}}$ for the second case.

However, the solution must lie on one of the three lines ($n = n_{\max}$, $n = n_{\min}$ or $k = f_{\min}T$), otherwise we would have found a minimum in the domain (we recall that the gradient of $E_{\mathrm{c,U}}(n,k)$ is never zero).

Let's focus on the first case, when $f_{0,\mathrm{U}} < f_{\min} < f_{1,\mathrm{U}}$ and evaluate $E_{\mathrm{c,U}}(n,k)$ on the $n = n_{\max}$ border. The problem is that the minimum $k_{0,\mathrm{U}} = \frac{s\alpha}{2n_{\max}}$ is now smaller than $f_{\min}T$. Since $E_{\mathrm{c,U}}(n,k)$ is decreasing for $k < k_{0,\mathrm{U}}$ and increasing for $k > k_{0,\mathrm{U}}$, the optimal point is $k = f_{\min}T$, which leads to the solution $(n_{\max}, f_{\min}T)$.

Similarly, let's look at the $k$ direction, evaluating the energy function on $k = f_{\min}T$. Now the minimum $n_{0,\mathrm{U}} = \frac{s(b+p)}{2pf_{\min}r}$ is beyond $n_{\max}$. Since $E_{\mathrm{c,U}}(n, f_{\min}T)$ is decreasing for $n < n_{0,\mathrm{U}}$ and increasing for $n > n_{0,\mathrm{U}}$ the optimal point is $n_{\max}$, which leads to the solution $(n_{\max}, f_{\min}T)$.

Eventually, now let's focus on the second case when $f_{\min} > f_{2,\mathrm{U}}$. We consider as same as the first case. We conclude that the solution is $(n_{\min}, f_{\min}T)$. Therefore we

finally conclude that the solution $(n, k)$ that gives the minimum energy consumption is:

$$\begin{cases} \text{if } f_{\min} \le f_{0,\mathrm{U}}, & S(n,k) = \left(n_{\max}, \dfrac{s\alpha}{2n_{\max}}\right) \\ \text{if } f_{0,\mathrm{U}} < f_{\min} < f_{1,\mathrm{U}}, & S(n,k) = (n_{\max}, f_{\min}T) \\ \text{if } f_{1,\mathrm{U}} \le f_{\min} \le f_{2,\mathrm{U}}, & S(n,k) = \left(\dfrac{(s_{\mathrm{b}}(b+p))}{(2pf_{\min}r)}, f_{\min}T\right) \\ \text{if } f_{\min} > f_{2,\mathrm{U}}, & S(n,k) = (n_{\min}, f_{\min}T) \end{cases} \tag{7.17}$$

With $f_{0,\mathrm{U}} = \frac{s_{\mathrm{b}}\alpha}{2n_{\max}}$, $f_{1,\mathrm{U}} = \frac{s_{\mathrm{b}}(b+p)}{2prn_{\max}}$ and $f_{2,\mathrm{U}} = \frac{s_{\mathrm{b}}(b+p)}{2prn_{\min}}$

In the remainder, we consider the other cases for data production PDFs.

### 7.2.2 Pareto Distribution

Here we assume that $P_{\mathrm{P}}(\psi_k)$ is a Pareto distribution with scale $v$ and $\alpha$, or:

$$P_{\mathrm{P}}(\psi_k) = \begin{cases} \alpha \dfrac{v^{\alpha}}{{\psi_k}^{\alpha+1}}, \psi_k \ge v \\ 0. \quad \text{otherwise} \end{cases} \tag{7.18}$$

The expected value of $\Psi$ is $E_{\mathrm{P}}[\psi_k] = \frac{\alpha v}{\alpha-1}$. Thus if we set $v = \frac{\alpha-1}{\alpha}kr$ we obtain $E_{\mathrm{P}}[\psi_k] = kr$, i.e., we match the expected transmission data to that of the Uniform distribution. In this case the energy expression in (7.2) becomes:

$$E_{\mathrm{c,P}} = ka + (g+p)\frac{\alpha v}{\alpha-1} + \frac{bs}{n} + (b+p)\left(\frac{v^{\alpha}n^{\alpha-1}}{a^{\alpha-1}(\alpha-1)} - \frac{\alpha v}{\alpha-1}\right) \tag{7.19}$$

With the same considerations as evaluated in the uniform case except there is no solution for $k$ direction. Therefore the solution $(n, k)$ giving the minimum energy consumption is:

$$\begin{cases} \text{if } f_{\min} < f_{1,\mathrm{P}}, & S(n,k) = (n_{\max}, f_{\min}T) \\ \text{if } f_{1,\mathrm{P}} \le f_{\min} \le f_{2,\mathrm{P}}, & S(n,k) = \left(\dfrac{\alpha s_{\mathrm{b}}}{(\alpha-1)f_{\min}r}\left(\dfrac{b}{(b+p)}\right)^{\frac{1}{\alpha}}, f_{\min}T\right) \\ \text{if } f_{\min} > f_{2,\mathrm{P}}, & S(n,k) = (n_{\min}, f_{\min}T) \end{cases} \tag{7.20}$$

With $f_{1,\mathrm{P}} = \frac{\alpha s_{\mathrm{b}}}{(\alpha-1)n_{\max}r}\left(\frac{b}{(b+p)}\right)^{\frac{1}{\alpha}}$ and $f_{2,\mathrm{P}} = \frac{\alpha s_{\mathrm{b}}}{(\alpha-1)n_{\min}r}\left(\frac{b}{(b+p)}\right)^{\frac{1}{\alpha}}$

### 7.2.3 Exponential Distribution

Considering $P_{\mathrm{E}}(\psi_k)$ as the exponential distribution with mean $E_{\mathrm{E}}[\psi_k] = kr$, we have:

$$P_{\mathrm{E}}(\psi_k) = \frac{1}{kr}\exp\left(-\frac{1}{kr}\psi_k\right) \tag{7.21}$$

Via the energy expression, we obtain:

$$E_{\mathrm{c,E}} = k[a + (g+p)r] + \frac{bs}{n} + (b+p)\left[kr(\exp\left(-\frac{a}{nkr}\right) - 1)\right] \tag{7.22}$$

With the same considerations as evaluated in the uniform case except there is no solution for $k$ direction. Therefore the solution $(n, k)$ giving the minimum energy consumption is:

$$\begin{cases} \text{if } f_{\min} < f_{1,\mathrm{E}}, & S\,(n,k) = (n_{\max}, f_{\min}\,T) \\ \text{if } f_{1,\mathrm{E}} \le f_{\min} \le f_{2,\mathrm{E}}, & S\,(n,k) = \left(\dfrac{s_{\mathrm{b}}}{f_{\min}\,r\,\ln\left(\frac{b+p}{b}\right)}, f_{\min}\,T\right) \\ \text{if } f_{\min} > f_{2,\mathrm{E}}, & S\,(n,k) = (n_{\min}, f_{\min}T) \end{cases} \tag{7.23}$$

With $f_{1,\mathrm{E}} = \frac{s_{\mathrm{b}}}{n_{\max}\,r\,\ln\left(\frac{b+p}{b}\right)}$ and $f_{2,\mathrm{E}} = \frac{s_{\mathrm{b}}}{n_{\min}\,r\,\ln\left(\frac{b+p}{b}\right)}$

### 7.2.4 Half-Gaussian Distribution

We start this part by considering $P_{\mathrm{H}}(\psi_k)$ as the Half-Gaussian distribution with mean $E_{\mathrm{H}}[\psi_k] = kr$.

$$P_{\mathrm{H}}(\chi_k) = \begin{cases} 0, & \psi_k < 0 \\ \dfrac{2}{\pi kr}\exp(\dfrac{{\psi_k}^2}{\pi k^2 r^2}), & \psi_k \ge 0 \end{cases} \tag{7.24}$$

Via the energy expression, we obtain:

$$\begin{aligned} E_{\mathrm{c,H}} &= k[a + (g+p)r] - \frac{ps}{n} + (b+p) \\ &+ \left[kr(\exp\left(-\frac{s^2}{\pi k^2 r^2 n^2}\right) - 1 + \frac{s}{n}\,\mathrm{erf}(\frac{s}{\sqrt{\pi}krn})\right] \end{aligned} \tag{7.25}$$

With the same considerations as evaluated in the uniform case except there is no solution for $k$ direction. Therefore the solution $(n, k)$ giving the minimum energy consumption is:

$$\begin{cases} \text{if } f_{\min} < f_{1,\mathrm{H}}, & S\,(n,k) = (n_{\max}, f_{\min}\,T) \\ \text{if } f_{1,\mathrm{H}} \le f_{\min} \le f_{2,\mathrm{H}}, & S\,(n,k) = \left(\dfrac{\alpha s_{\mathrm{b}}}{\sqrt{\pi} f_{\min}\,r\,\mathrm{erf}^{-1}\left(\frac{p}{b+p}\right)}, f_{\min}\,T\right) \\ \text{if } f_{\min} > f_{2,\mathrm{H}}, & S\,(n,k) = (n_{\min}, f_{\min}T) \end{cases} \tag{7.26}$$

With $f_{1,\mathrm{H}} = \frac{\alpha s_{\mathrm{b}}}{\sqrt{\pi} n_{\max}\,r\,\mathrm{erf}^{-1}\left(\frac{p}{b+p}\right)}$ and $f_{2,\mathrm{H}} = \frac{\alpha s_{\mathrm{b}}}{\sqrt{\pi} n_{\min}\,r\,\mathrm{erf}^{-1}\left(\frac{p}{b+p}\right)}$

## 7.3 Evaluation of the Analytic Results

To validate the proposed analytical model we performed a series of experiments on a typical WSN platform, namely the TelosB sensor nodes (equipped with the IEEE802.15.4 compliant CC2420 radio transceiver and running the low-power Contiki 2.6 operating system). We implemented a sensor network operated by the recently proposed TFDMA protocol, which allows for collision-free transmission. Hence, we could utilize the very low-complexity NullMAC and NullRDC options of the Contiki RTOS, which lead to a maximum data consumption rate at the application layer of $s_{\mathrm{b}} = 144$ kbps. To simulate the data production process, we artificially created data according to the PDFs considered in previous section via rejection sampling [59], setting the mean transmission rate to $r = 5.2$ kbit. The data is copied onto each sensor node during deployment, and it is read at runtime and considered as coming from the multimedia processor.

| Parameter | Unit | Value |
|---|---|---|
| $s_{\mathrm{b}}$ | kbps | 144 |
| $r$ | kbit | 5.2 |
| $a$ | J | $4.000 \times 10^{-3}$ |
| $g$ | J/b | $2.197 \times 10^{-7}$ |
| $b$ | J/b | $1.902 \times 10^{-7}$ |
| $p$ | J/b | $2.861 \times 10^{-7}$ |
| $n_{\min}, n_{\max}$ | - | 2, 16 |
| $f_{\min}$ | fps | 2 |

Table 7.1: System Settings

Under these operational settings, we measured the real-time energy consumption of each sensor node by integrating its instantaneous current consumption profile, which we obtained using a Tektronix MDO4104-6 oscilloscope and measuring the current consumption at a high-tolerance 1 Ohm resistor placed in series with each TelosB node. Under this setup, we also measured the different energy costs for transmission, beaconing and buffering, which are reported in Table 7.1. The cost of acquiring an image is derived from the

specifications of the OV7670 camera sensor, which is widely used in low-power visual sensor platforms, such as the one proposed in [134] [135]. To simulate the image acquisition process, the energy consumption measured for a particular value of $k$ is increased artificially by $ka$ Joules.

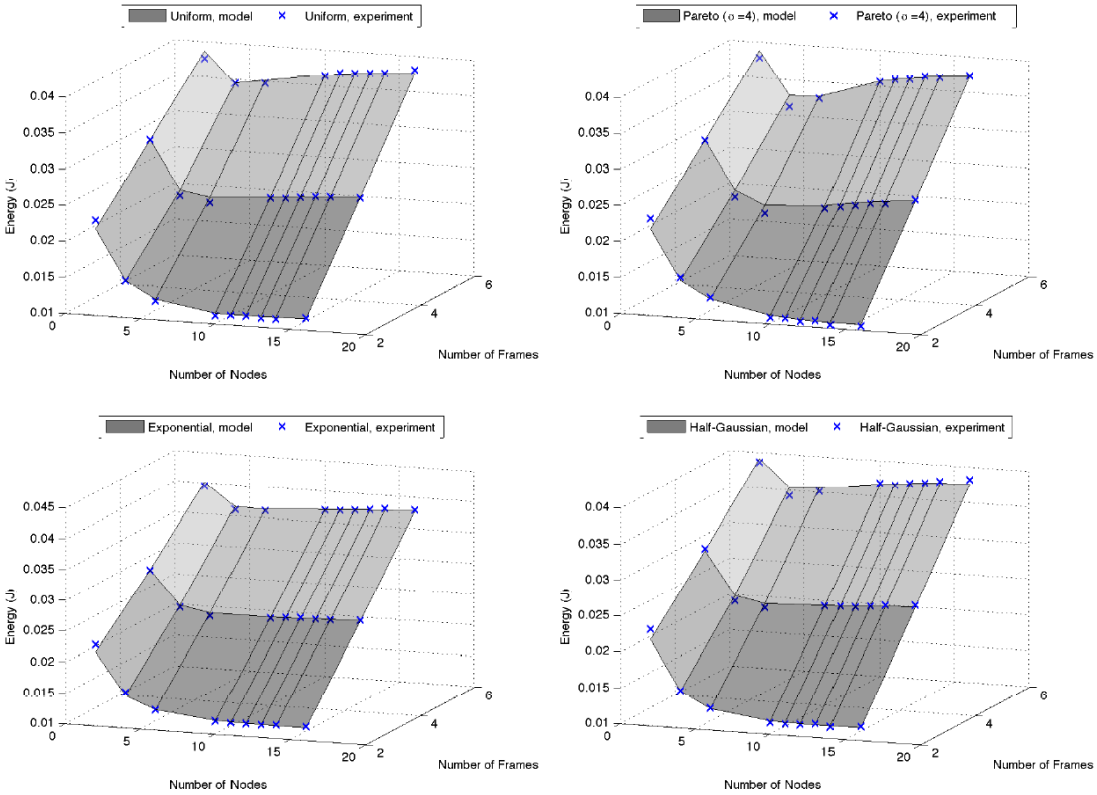


Figure 7.1: The grayscale surfaces show, for each statistical distribution, the energy consumption of a single camera sensor node as a function of the frame rate and the total number of nodes in the TDMA schedule. The blue crosses correspond to the value of the consumed energy as measured from the sensor network testbed correspond to active time of the sensor node in 1s.

| **Transmission Rate PDFs** | **Mean Error (%)** | **Max Error (%)** | **$R^2$ value** | **Theoretical Optimum** |
|---|---|---|---|---|
| Uniform | 1.25 | 5.24 | 0.9982 | $n = 12, k = 2$ |
| Pareto ($\alpha = 4$) | 1.90 | 6.59 | 0.9988 | $n = 14, k = 2$ |
| Exponential | 1.00 | 5.35 | 0.9977 | $n = 16, k = 2$ |
| Half-Gaussian | 1.33 | 6.65 | 0.9963 | $n = 13, k = 2$ |

Table 7.2: Differences between the theoretical and experimental results and the optimal value of the nodes number and the frames number (on the last column) amongst the considered PDF under the settings of Figure 7.1.

Then, for each data production PDF considered in the previous section, we computed the theoretical energy function and the optimal values of $k$ and $n$ that

allow for minimum energy consumption, using the results in (7.17),(7.20),(7.23) and (7.26). As one can see from Figure 7.1, for all the tested distributions the theoretical and experimental results are in agreement, with the maximum percentage error between them limited to 6.65%. We have observed the same level of accuracy under a variety of TFDMA settings and minimum frame rate $f_{\min}$ but omit these repetitive experiments for brevity of exposition.

## 7.4 Applications

So far, we have considered four different distributions for the data generation process. However, it is interesting to study a real-case scenario, in order to assess that the proposed optimization model can be applied in an actual visual sensor network deployment. In this section, we consider two different multimedia applications, namely *(i)* encoding and transmission of JPEG frames and *(ii)* extraction and transmission of corner-like features for visual analysis. For the input data, we considered the video sequences from the PETS2007 dataset[10], which are taken from an airport surveillance video system. The resolution of all sequences is 768 x 576 pixels and the frame rate is 25 fps.

1. ***JPEG compression***: We simulated a hybrid DCT-DPCM system, such as the one presented in [136]. In this system, the first frame of the video sequence is JPEG encoded and transmitted. For the subsequent frames, only the difference between two adjacent frames is encoded. The encoding process follows the standard JPEG baseline, i.e., quantization of the Discrete Cosine Transform (DCT) coefficients followed by run length encoding and Huffman coding. The resulting encoded frame size is stored in a file, which is then fed to the sensor network testbed for energy measurements. The process is repeated for different values of $k$ by reducing the input video sequence frame rate.

2. ***Visual feature extraction***: Several visual analysis tasks can be performed by disregarding the pixel representation of an image, and relying only on a much

[10] http://pets2007.net

more compact representation based on local visual features [137]. In a nutshell, salient keypoints of an image are identified by means of a detector, and a descriptor is computed from the pixel values belonging to the image patch around the keypoint. Here, we focus on corner-like local features. Precisely, we process each frame of the input video sequence with the FAST corner detector [138], which is optimized for fast extraction of visual features on low-power devices. Each detected keypoint is then described by means of a binary descriptor, such as BRIEF [139], where each element of the descriptor itself is a bit that reports the result of an intensity comparison between two pixels of the patch to be described. Here we assume that each descriptor is composed by 512 binary tests, for a descriptor size of 64 bytes. Thus, each frame will require $64m$ bytes to be transmitted, being $m$ the number of keypoints detected by the FAST algorithm. Also in this case, the process is repeated for different video sequences frame rate.

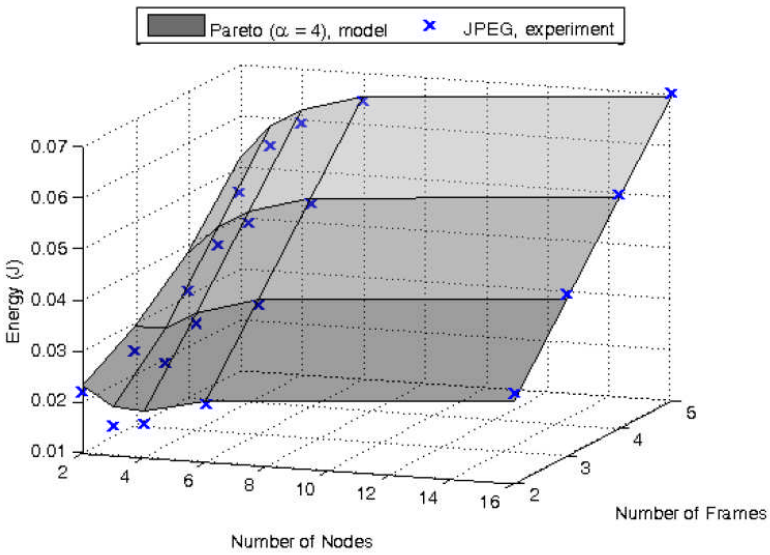


Figure 7.2: Energy consumption for JPEG application, the grayscale surfaces represent the fitted energy function obtained with the Pareto PDF equation, while the blue crosses represent the experimental measurements correspond to the active time of the sensor node in 1s.

We repeated the experimental measurements described in previous section for both application scenarios. Then, we fitted[11] the energy measurements with one of

[11] Fitting is performed by matching the average data size $r$ of each distribution to the average data size of the JPEG compressed frames and the set of visual features.

the energy functions derived in Section 7.2. Interestingly, we found that assuming the data samples for both the application scenario as coming from a Pareto distribution produces a good fit, as shown in Figure 7.2 and Figure 7.3, with a coefficient of determination value $R^2$ of 0.96 for the JPEG case and 0.95 for the salient point case. Thus, we can utilize the results for the minimum energy consumption in the Pareto case to discuss the gain that can be potentially achieved by following the proposed approach. As an example, in Table 7.3, we consider two different cases for each application scenario. Each case has different application constraints and we compare the optimal solution with an ad-hoc "conservative" solution obtained by choosing the minimum frame rate and the minimum and maximum number of nodes allowable. As one can see, the proposed approach allows to obtain significantly energy savings, as high as 45.65%.

| Case | Constraints | JPEG compression: $r$ = 20.5 kbit | | | Visual feature extraction: $r$ = 11.7 kbit | | |
|---|---|---|---|---|---|---|---|
| | | Ad-hoc deployment | Proposed approach | Gain (%) | Ad-hoc deployment | Proposed approach | Gain (%) |
| I | $f_{\min} = 0.2$ fps<br>$n_{\min} = 3$<br>$n_{\max} = 6$ | $n = 3$<br>$k = 0.2$<br>$E_c = 0.010$ J | $n = 6$<br>$k = 0.2$<br>$E_c = 0.005$ J | 45.41 | $n = 3$<br>$k = 0.2$<br>$E_c = 0.010$ J | $n = 6$<br>$k = 0.2$<br>$E_c = 0.005$ J | 46.65 |
| II | $f_{\min} = 2$ fps<br>$n_{\min} = 2$<br>$n_{\max} = 10$ | $n = 2$<br>$k = 2$<br>$E_c = 0.023$ J | $n = 4$<br>$k = 2$<br>$E_c = 0.019$ J | 17.76 | $n = 2$<br>$k = 2$<br>$E_c = 0.022$ J | $n = 6$<br>$k = 2$<br>$E_c = 0.014$ J | 35.96 |

Table 7.3: Minimum energy consumption under ad-hoc settings and proposed framework. The energy saving shows in the percentile difference between the ad-hoc and proposed cases for two sample application scenarios.

## 7.5 Conclusions

We applied an analytic framework for characterizing practical energy consumption in uniformly-formed network with the case of the visual sensor networks. We focused on the case where the network is operated by collision-free TDMA and deployed to carry out a delay-tolerant monitoring task. This framework was solved to optimality for different multimedia traffic distribution and validated through a real sensor network testbed with accuracy below 7% of the energy consumption. Finally, we applied this model to two real case scenarios, demonstrating that

substantial energy saving can be obtained with the proposed approach, with respect to an ad hoc system design.

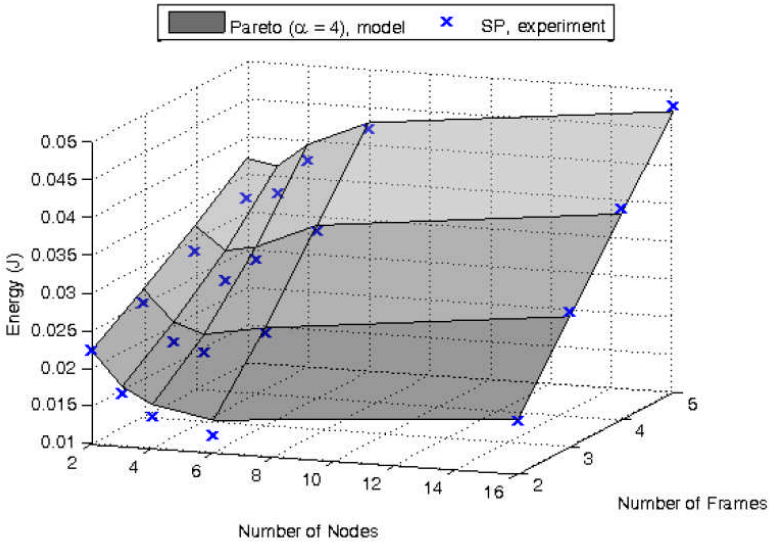


Figure 7.3: Energy consumption for salient point application, the grayscale surfaces represent the fitted energy function obtained with the Pareto PDF equation, while the blue crosses represent the experimental measurements correspond to active time of the sensor node in 1s.

# Chapter 8
# Conclusion

This work studies MAC-layer distributed (de)synchronization for efficient TDMA mechanisms in real-world WSNs, with a focus on the IEEE802.15.4 MAC. We reviewed the literature behind of all well-established synchronization approaches in WSNs. It consists of proposals for global-clock–based synchronization and local-clock–based synchronization. Each of them is separated into centralized and distributed synchronization. In this thesis we focus only the distributed synchronization with working on its own clock.

In Chapter 2, we review, implement and evaluate the basic algorithms proposed in the literature for distributed TDMA in WSNs based on synchronization and desynchronization. Amongst the three proposals reviewed, the SYNC algorithm has the highest implementation complexity due to the reach-back response and the complexity in managing the slot time. The DESYNC algorithm achieves the highest network resource utilization and has the lowest implementation complexity, with the PCO-DESYNC achieving similar but slightly inferior results.

In Chapter 3, we focus on these two algorithms (DESYNC and PCO-DESYNC) and characterize the convergence time via stochastic modelling and assuming a stochastic convergence criterion. The results of the experimental validation with varying coupling parameter $\alpha$ and different convergence threshold $b_{\text{thres}}$ agree with the results of the mathematical model for both algorithms. Under high threshold for convergence, the two algorithms achieve higher bandwidth and lower delay to steady state when compared under low threshold for convergence. As a side effect of our analysis, our model analytically establishes the coupling parameter that minimizes the expected iterations required until convergence under both algorithms.

In Chapter 4, we extend the desynchronization primitive to multi-hop WSN communications. The hidden node problem has been handled via the multi-hop

extension of the basic DESYNC algorithm and experimental results show that the proposed solution is simple and efficient in dealing with arbitrary multi-hop topologies in WSNs.

In Chapter 5, we focus on the multi-channel extension of DESYNC to achieve increased bandwidth efficiency in a distributed (coordinator-less) manner. This new protocol is called the TFDMA. We propose an analytic model for estimating the expected delay to balanced state and prove that TFDMA is stable. The experimental results agree with the theoretical model for the convergence delay and indeed demonstrate the increased bandwidth efficiency of TFDMA against single-channel DESYNC and two other state-of-the-art multi-channel WSN protocols.

In Chapter 6, we analyze the energy consumption for distributed TDMA with various transmission rate PDFs in order to support various applications producing irregular data payloads. We found the optimal number of nodes achieving the minimum energy consumption under a uniformly-formed WSN fully-connected topology. Our results show that the derived analytic formulation of the energy consumption agrees with an experimental test-bed based on TelosB motes.

In Chapter 7, we extend the analytic formulation of energy consumption of distributed TDMA to the case of visual sensor networks where competing constraints are set with respect to the required frame-rate per sensor and the total number of sensors to deploy (spatial versus temporal coverage in the WSN). Two examples of multimedia applications are used for experimental validation: JPEG compression and salient point extraction from video frames. We found that the data processing and transmission rate of the JPEG and salient point case can be well approximated with the Pareto distribution.

## 8.1 Future Work

In the future work, we suggest to study further the optimized space-time-frequency mechanisms for distributed (de)synchronization in WSNs. The aim is to establish the best approach for *distributed* desynchronization and bandwidth

efficiency while at the same time allowing for energy-efficient and scalable operation within a WSN infrastructure based on IEEE802.15.4 MAC.

This goal of future work is an energy-efficient, high-throughput *distributed* multi-channel communication scheme for the main desynchronization primitives (DESYNC and PCO-DESYNC) realized on a low-resource platform with IEEE802.15.4 compatibility. In [24], high communication bandwidth can be achieved if the load in wireless network is balanced. The authors propose a cognitive load balance algorithm for single-hop multi-channel sensor networks. However, like most work on this topic [25][26], all such balancing schemes are related to the centralized case, where an unfaltering (coordinating) node is present and manages the entire process. This type of base station or sink node is responsible for broadcasting the control message to allocate the channel utilization to each node in the network. However, this is inappropriate for WSNs deployed under completely unpredictable conditions, e.g. for surveillance or monitoring at remote and potentially hostile environments. To this end, this work is related to the *decentralized* WSN case.

We are also interested in time synchronized channel hopping, which has been proposed recently as an extension of the standard IEEE 802.15.4e for the case of decentralized WSNs [30][31]. Within the context of this work item, we can devise a theoretical and experimental comparison between our scheme (time-frequency distributed desynchronization) and the decentralized TSCH. There are many interesting metrics for evaluation such as energy consumption, connectivity [113][114], throughput, as well as message loss.

Eventually, a related task is the proposed scheme of multi-hop based synchronization primitives [108][109]. Initial proposals for distributed time synchronization protocols within multihop WSNs exist in the literature [27][28]. They developed efficient communication protocols based on two-hop information. Degesys *et al* [15] also discuss extensions of the DESYNC algorithm towards multi-hop networks using simulations. Overall, there is sufficient interest from the research community in this topic to warrant the study of all synchronization

primitives for the case of multi-hop networks, especially via the use of real WSN-based implementations. Combining of the multiple-channel and multi-hop protocols proposed is a challenging task that could be attempted within future work in this field.

# A. Appendix 1

We elaborate further why the central limit theorem is applicable within the DESYNC and PCO formulation.

All initial random variables, i.e. vectors $\Phi^{(0)}$ and noise vector $\Delta$ (the latter is inserted at each phase update iteration) are independent, following Definition 2 and Definition 3. At each phase update iteration $k$, each updated phase random variable $\Phi_i^{(k)}$ is the sum of independent, random variables: $\Phi_{i-1}^{(0)}$, $\Phi_i^{(0)}$, $\Phi_{i+1}^{(0)}$,... and noise terms: $\forall l \in \{0, \dots, k-1\}$: $\Delta_{i-1}^{(l)}$, $\Delta_i^{(l)}$, $\Delta_{i+1}^{(l)}$,... scaled by constants of the form:

$$\forall l\colon (1-\alpha)^l, \left(\frac{\alpha}{2}\right)^l, (1-\alpha)\left(\frac{\alpha}{2}\right)^l, (1-\alpha)^l\left(\frac{\alpha}{2}\right), \text{etc.}$$

This sum grows at each iteration due to new independent noise and phase terms being inserted. For example, for DESYNC we have:

At 1st phase update iteration;

$$\Phi_i^{(1)} = \left[(1-\alpha)(\Phi_i^{(0)} + \Delta_i^{(0)}) + \alpha\frac{\Phi_{i-1}^{(0)}+\Delta_{i-1}^{(0)}+\Phi_{i+1}^{(0)}+\Delta_{i+1}^{(0)}}{2} \pmod 1\right]$$

Then 2nd phase update iteration;

$$\Phi_i^{(2)} = \left[(1-\alpha)(\Phi_i^{(1)} + \Delta_i^{(1)}) + \alpha\frac{\Phi_{i-1}^{(1)}+\Delta_{i-1}^{(1)}+\Phi_{i+1}^{(1)}+\Delta_{i+1}^{(1)}}{2} \pmod 1\right]$$

$$= \left[(1-\alpha)^2(\Phi_i^{(0)} + \Delta_i^{(0)}) + (1-\alpha)\alpha\frac{\Phi_{i-1}^{(0)}+\Delta_{i-1}^{(0)}+\Phi_{i+1}^{(0)}+\Delta_{i+1}^{(0)}}{2} + \frac{\alpha}{2}\left[(1-\alpha)\Phi_{i-1}^{(0)} + \alpha\frac{\Phi_{i-2}^{(0)}+\Delta_{i-2}^{(0)}+\Phi_i^{(0)}+\Delta_i^{(0)}}{2} + (1-\alpha)\Phi_{i+1}^{(0)} + \alpha\frac{\Phi_i^{(0)}+\Delta_i^{(0)}+\Phi_{i+2}^{(0)}+\Delta_{i+2}^{(0)}}{2}\right] + (1-\alpha)\Delta_i^{(1)} + \alpha\frac{\Delta_{i-1}^{(1)}+\Delta_{i+1}^{(1)}}{2} \pmod 1\right]$$

$\Phi_i^{(3)} = \cdots$ (3rd phase update iteration)

$\vdots$

$\Phi_i^{(k)} = \cdots$ ($k^{\text{th}}$ phase update iteration)

We can derive similar expressions for PCO-based desynchronization albeit with the difference that only noise terms are inserted in each iteration (and no new phase terms):

At 1st phase update iteration;

$$\Phi_i^{(1)} = [(1-\alpha)(\Phi_i^{(0)} + \Delta_i^{(0)}) + \alpha(1-\tfrac{1}{W}) \pmod 1]$$

Then 2nd phase update iteration;

$$\Phi_i^{(2)} = [(1-\alpha)(\Phi_i^{(1)} + \Delta_i^{(1)}) + \alpha(1-\tfrac{1}{W}) \pmod 1]$$

$$= [(1-\alpha)^2(\Phi_i^{(0)} + \Delta_i^{(0)}) + (1-\alpha)\Delta_i^{(1)} + (2-\alpha)\alpha(1-\tfrac{1}{W}) \pmod 1]$$

$$\Phi_i^{(3)} = \cdots \text{ (3rd phase update iteration)}$$

$$\vdots$$

$$\Phi_i^{(k)} = \left[(1-\alpha)^k \Phi_i^{(0)} + \sum_{j=1}^{k}(1-\alpha)^j \Delta_i^{(k-j)} + \alpha\left(1-\tfrac{1}{W}\right)\sum_{j=0}^{k-1}(1-\alpha)^j \pmod 1\right]$$

[$k^{\text{th}}$ phase update iteration – this is actually (3.28)], with $\Delta_i^{(k-j)}$ iid random variables, each stemming from Definition 3. As a result, each term $\Phi_i^{(1)}, \Phi_i^{(2)}, \ldots, \Phi_i^{(k)}$ comprises the sum of independent (but not identically-distributed) random variables[12].

According to Papoulis [ref. [14] pp. 219-220], a set of sufficient conditions for the CLT to hold, i.e. $\Phi_i^{(k)} \xrightarrow[k\to\infty]{} N(\mu_{\Phi_i^{(k)}}, \sigma_{\Phi_i^{(k)}})$, is:

*(a)* $\Phi_i^{(k)}$ must be derived as the sum of independent but not necessarily identically-distributed random variables

*(b)* A constant $\varepsilon > 0$ exists such that $\forall k$: $\sigma_{\Phi_i^{(k)}} > \varepsilon$.

*(c)* $\forall i, k$: densities $\mathrm{P}_{\Phi_i^{(k)}}$ are zero outside a finite interval no matter how large the interval may be.

Condition *(a)* holds as elaborated above. Condition *(b)* holds since, $\forall k \in \mathbb{N}^*$, we can pick $\varepsilon = (1-\alpha)^k \sigma_{\Phi^{(0)}}$ (that satisfies the requirement that $\varepsilon > 0$) and then, from (3.15): $\sigma_{\text{desync}}^{(k)} > \varepsilon$ and, from (3.23): $\sigma_{\text{PCO}}^{(k)} > \varepsilon$. Finally, condition *(c)* holds for our case because all initial PDFs have finite support (they are all variants of the uniform distribution); hence, densities $\mathrm{P}_{\Phi_i^{(k)}}$ will have finite support since they are linear mixtures of PDFs with finite support.

As a final note, Papoulis remarks [ref. [14] pp. 215] that if the PDFs in the mixture are smooth, the convergence to the normal distribution is fast, i.e. even the sum of five random variables can be assumed to be normally distributed with very good approximation accuracy.

[12] We note that scaling and shifting a random variable by a constant will change its moments by stretching and shifting its distribution function, but it will not make it dependent to other random variables.

# B. Bibliography


[1] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," *Proc. of the 2nd ACN conf. on Embedded Networked Sensor Systems*, Nov. 2004.

[2] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 214-226, Feb. 2006.

[3] K. Marzullo and S. Owicki, "Maintaining the time in a distributed system," *Proc. of the 2nd Ann. ACM Symp. on Principles of Distributed Computing*, vol. 19, no. 3, Jul .1985.

[4] R. Leidenfrost and W. Elmenreich, "Firefly clock synchronization in an 802.15.4 wireless network," *EURASIP J. on Embedded Syst.*, vol. 2009, Jan. 2009.

[5] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing –sync protocol for sensor networks," *Proc. of the 1st Int. Conf. on Embedded Networked Sensor Syst.*, Nov. 2003.

[6] S. Ganeriwal, R. Kumar, S. Adlakha, and M. Srivastava, "Network-wide time synchronization in sensor networks," NESL, Tech. Rep., 2003.

[7] J. Elson, L. Firod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *Proc. of the 5th Symp. on Operating Systems Design and Implementation*, vol. 36, Dec. 2002.

[8] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "DESYNC: Self-organizing desynchronization and TDMA on wireless sensor networks," *6th Int. Symp. on IPSN,* Apr. 2007.

[9] R. Pagliari, Y.-W. P. Hong, and A. Scaglione, "Bio-inspired algorithms for decentralized round-robin and proportional fair scheduling," *IEEE J. on Select. Areas in Commun.,* vol. 28, no. 4, pp. 564-575, May 2010.

[10] Y. W. Hong and A. Scaglione, "A scalable synchronization protocol for large scale sensor networks and its applications," *IEEE J. on Selected Areas in Communications,* vol. 23, no. 5, pp. 1085-1099, May. 2005.

[11] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," *Proc. of the 3rd Int. Conf. on Embedded Networked Sensor Syst.*, Nov. 2005.

[12] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz, "Distributed synchronization in wireless networks," *IEEE Signal Process. Mag.,* vol. 25, no. 5, pp. 81-97, Sept. 2008.

[13] A. Patel, J. Degesys, and R. Nagpal, "Desynchronization: The theory of self-organizing algorithms for round-robin scheduling," *Proc. IEEE Internat. Conf. on Self-Adaptive and Self-Organizing Syst. (SASO),* pp. 87-96, Jul. 2007.

[14] A. Papoulis, Probability and Statistics, Prentice Hall, 1989.

[15] J. Degesys and R. Nagpal, "Towards desynchronization of multi-hop topologies," *Second IEEE Int. Conf. on Self-Adaptive and Self-Organizing Syst.*, Oct. 2008.

[16] M. Ramakrishnan and P. V. Ranjan, "Multi channel MAC for wireless sensor networks," *Int. J. of Computer Networks & Communications,* vol. 1, no. 2, pp. 47-54, Jul. 2009.

[17] Crossbow Technology, Inc., "Imote2.builder kit manual," rev. A, Sep. 2007

[18] A. Motskin. T. Roughgarden, P. Skraba, and L. Guibas, "Lightweight coloring and desynchronization for networks," *IEEE Infocom*, pp. 2389-2391, Apr. 2009.

[19] D. Buranapanichkit and Y. Andreopoulos, "Stochastic modelling of convergence to desynchronization primitive in wireless sensor networks," *Submitted to ACM Transactions on Autonomous and Adaptive System*, - Under review.

[20] M. L. Sichitiu and C. Veerarittiphan, ''Simple, accurate time synchronization for wireless sensor networks,'' *Proc. IEEE Wireless Communications and Networking,* vol. 2, pp. 1266–1273, Mar. 2003,

[21] J. Van Greunen and J. Rabaey, ''Lightweight time synchronization for sensor networks,'' *Proc. 2nd ACM Int. Conf. Wireless Sensor Networks and Applications (WSNA),* pp. 11–19, 2003,

[22] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, ''Clock synchronization for wireless sensor networks: a survey,'' *Ad-Hoc Networks*, vol. 3, no. 3, pp. 281–323, Mar. 2005.

[23] Y. C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Process. Mag.,* vol. 28, no. 1, pp. 124-138, Jan. 2011.

[24] M. Song, Y. Zhao, J, Wang, and Park, E.K., "A high throughput load balance algorithm for multichannel wireless sensor networks,"*ICC '09. IEEE International Conference*, pp. 1-5, Jun. 2009.

[25] Y. Zeng, N. Xiong, and T. Kim, "Channel assignment and scheduling in multichannel wireless sensor networks," *33rd IEEE Conference on Local Computer Networks,* pp. 512-513,Oct. 2008

[26] Y. Kim, H. Shin, and H. Cha, "Y-MAC: An energy-efficient multi-channel MAC protocol for dense wireless sensor networks," *International Conference on Information Processing in Sensor Networks,* Apr. 2008.

[27] R. Solis, V. Borkar and P. R. Kumar, "A new distributed time synchronization protocol for multihop wireless networks." *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 2734-2739, Dec. 2006.

[28] K.-Y. Cheng, K.-S. Lui, Y.-C. Wu, and V. Tam, ''A distributed multihop time synchronization protocol for wireless sensor networks using pairwise broadcast synchronization,'' *IEEE Trans. Wireless Commun.,* vol. 8, no. 4, pp. 1764–1772, Apr. 2009.

[29] http://www.ieee802.org/15/pub/TG4e.html

[30] A. Tinka, T. Watteyne, and K. Pister "A decentralized scheduling algorithm for time synchronized channel hopping," *Lect. Notes Inst. Comp. Sci. (Proc. Ad-Hoc Networks),* vol. 49, no. 4, pp. 201-216, Aug.2010.

[31] T. Watteyne, S. Lanzisera, A. Mehta, and K. Pister, "Mitigating multipath fading through channel hopping in wireless sensor networks", *IEEE International Conference on Communications (ICC 2010),* pp 1-5, May 2010

[32] K. Pister and L. Doherty, "TSMP: Time synchronized mesh protocol," *Parallel and Distributed Computing and Systems Proc. IASTED Symp. On Distr. Sensor Netw.,* pp. 391-398, Nov. 2008.

[33] IEEE 802.15.4 Standard-2003, "Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)," *IEEE-SA Standards Board*, 2003.

[34] D. Buranapanichkit and Y. Andreopoulos, "Distributed time-frequency division multiple access protocol for wireless sensor networks," *IEEE Wireless Comm. Letters*, vol. 1, no. 5, pp. 440-443, Oct 2012.

[35] A. Koubaa and M. Alves, "An IEEE 802.15.4 protocol implementation (in nesC/TinyOS): reference guide v1.2," *IPP-HURRAY Technical Report, HURRAY-TR-061106*, Feb. 2007.

[36] R. Pagliari, Y.-W. P. Hong, and A. Scaglione, "Pulse coupled oscillators' primitives for collision-free multiple access with application to body area networks," *IEEE Applied Sciences on Biomedical and Communication Technologies, ISABEL*, pp. 1-5, Oct, 2008.

[37] H. K. Le, D. Henriksson, and T. Abdelzaher, "A practical multi-channel media access control protocol for wireless sensor networks," *Proc. IEEE IPSN,* pp. 70-81, 2008.

[38] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," *Proc. Mobicom'05.*

[39] X. Lin and S. B. Rasool, "Distributed and provably efficient algorithms for joint channel-assignment, scheduling, and routing in multichannel ad hoc wireless networks," *IEEE/ACM Trans. Netw.,* vol. 17,no. 6, Dec. 2009.

[40] A. Koubaa, M. Alves and E. Tovar, "GTS allocation analysis in IEEE 802.15.4 for real-time wireless sensor networks," *Proc. IEEE 20th Int. Symp. On IPDPS,* Apr.2006.

[41] P. Jurcik, A. Koubaa, M. Alves, E. Tovar, Z. Hanzalek, "A simulation model for the IEEE 802.15.4 protocol: delay/throughput evaluation of the GTS mechanism," *Proc. IEEE MASCOTS,* 2007.

[42] P. Park, C. Fischione and K. H. Johansson, "Performance analysis of GTS allocation in beacon enabled IEEE 802.15.4" *Proc. IEEE Secon,* Jun. 2009.

[43] V. G. Kulkarni, Modeling and Analysis of Stochastic Systems, Chapman Hall, London 1995.

[44] L. Tang, Y. Sun, O. Gurewitz, D. B. Johnson, "EM-MAC: A dynamic multichannel energy-efficient MAC protocol for wireless sensor networks," *Proc. 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc,* 2011.

[45] http://www.ee.ucl.ac.uk/~iandreop/WCL2012.zip

[46] J. Song, S. Han, A.K. Mok, D. Chen, M. Lucas, and M. Nixon, "WirelessHART: Applying wireless technology in real-time industrial process control," *IEEE Real-Time and Embed. Tech. And Appl. Symp.,2008 RTAS'08.* IEEE, pp. 377-386, 2008.

[47] A. Koubaa, A. Cunha, and M. Alves, "A time division beacon scheduling mechanism for IEEE 802.15.4/zigbee cluster-tree wireless sensor networks," *19th Euromicro Conf. on Real-Time Syst., ECRTS'07,* pp. 125-135, Jul. 2007.

[48] J. Kho, A. Rogers, and N.R. Jennings, "Decentralized control of adaptive sampling in wireless sensor networks," *ACM Trans. On Sensor Netw.,* vol. 5, no. 3, pp. 19, 2009.

[49] A. McCree, K. Brady, and T.F. Quatieri, “Mulitsensor very lowbit rate speech coding using segment quantization,” *IEEE Internat. Conf. on Acoust., Speech and Signal Proc.* IEEE, pp. 3997-4000, 2008.

[50] M. Pursley and L. Davisson, “Variable rate coding for nonergodic sources and classes of ergodic sources subject to a fidelity constraint,” *IEEE Trans. On Inf. Theory,* vol. 22, no. 3, pp. 324-337, 1976.

[51] A. Koubaa, M. Alves, M. Attia, and A. Van Nieuwenhuyse, “Collision-free beacon scheduling mechanisms for IEEE 802.15.4/zigbee cluster-tree wireless sensor networks,” *Proc. of the 7th Int. Worksh. On Appl. and Serv. In Wireless Netw. (ASWN),* 2007.

[52] V. Paxson and S. Floyd, “Wide area traffic: the failure of Poisson modelling,” *IEEE/ACM Trans. On Networking,* vol. 3, no. 3, pp. 226-244, Mar. 1995.

[53] K. Park, G. Kim, and M. Crovella, “On the relationship between file sizes, transport protocols, and self-similar network traffic,” *Proc. 1996 Int. Conf. on Network Prot.* IEEE, pp. 171-180, 1996.

[54] M. Dai, Y. Zhang, and D. Loguinov, “A unified traffic model for MPEG-4 and H.264 video traces,” *IEEE Trans. On Multimedia,* vol. 11, no. 5, pp. 1010-1023, May 2009.

[55] M. Tagliasacchi, S. Tubaro, and A. Sarti, “On the modeling of motion in Wyner-Ziv video coding,” *IEEE Int. Conf. on Image Process.* IEEE, pp. 593-596, 2006.

[56] E.Y. Lam and J.W. Goodman, “A mathematical analysis of the DCT coefficient distributions for images,” *IEEE Trans. On Image Process.,* vol. 9, no. 10, pp. 1661-1666, Oct. 2000.

[57] Y. Andreopoulos and I. Patras, “Incremental refinement of image salient-point detection,” *IEEE Trans. On Image Process.,* vol. 17, no. 9, pp. 1685-1699, 2008.

[58] G. Lu, B. Krishnamachari, and C.S. Raghavendra, “Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks,” *IEEE Int. Conf. on Perf., Comput., and Comm.* IEEE, pp. 701-706, 2004.

[59] W.R. Gilks and P. Wild, “Adaptive rejection sampling for Gibbs sampling,” *Applied Statistics*, pp. 337-348, 1992.

[60] Crossbow Technology, Inc., “Imote2 hardware reference manual,” rev. A, Sep. 2007

[61] Moteiv Corporation, “Telos (Rev B): PRELIMINARY Datasheet,” May, 2004.

[62] T. Nagayama and B.F. Spencer, Jr., "Structural health monitoring using smart sensors," NSEL Report, No. NSEL-001, Nov. 2007.

[63] J. Mo, H. S. Wilson and J. Walrand, "Comparison of multichannel MAC protocols," *IEEE Trans. on Mob. Comput.,* vol. 7, no. 1, pp. 50-65, Jan, 2008.

[64] G. Zhou, C. Huang, T. Yan, T. He and J. A. Stankovic, "MMSN: Multi-Frequency media access control for wireless sensor networks," *IEEE InfoCom.* 2006.

[65] H.W. So, J. Walrand and J. Mo, "McMAC: A parallel rendezvous multi-channel MAC protocol," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC'07),* Mar. 2007.

[66] J. So and N. Vaidya, "Multi-Channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single transceiver," *Proc. ACM MobiHoc,* May 2004.

[67] T. Luo, M. Motani and V. Srinivasan, "Cooperative asynchronous multichannel MAC: design, analysis, and implementation," *IEEE Trans. on Mob. Comput.,* vol. 8, no. 3, Mar 2009.

[68] S.-L. Wu, Y.-C. Tseng, C.-Y. Lin and J.-P. Sheu, "A multi-channel MAC protocol with power control for multi-hop mobile ad hoc networks," *The Computer J.,* vol. 45, no. 1, pp. 101-110, 2002

[69] W.-C. Hung, K.L.E. Law and A. Leon-Garcia, "A dynamic multi-channel MAC for ad hoc LAN," *Proc. 21st Biennial Symp. On Comm.,* Apr. 2002.

[70] A. Kansal, J. Hsu, S. Zahedi and M.B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embed. Comput. Syst.,* vol. 6, no. 4, Sep. 2007

[71] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: survey and implications," *IEEE Comm. Surv. Tut.,* vol. 13, no. 3, pp. 443-461, quarter 2011.

[72] S. Chalasani and J.M. Conrad, "A survey of energy harvesting sources embedded systems," *IEEE Southeastcon, 2008*, pp.442-447, Apr. 2008

[73] V. Sharma, U. Mukherji, V. Joseph and S. Gupta, "Optimal energy management policies for energy harvesting sensor nodes," *IEEE Trans. On Wireless Comm.,* vol. 9, no. 4, pp. 1326-1336, Apr. 2010.

[74] B. Zhang, R. Simon and H. Aydin, "Energy management for time critical energy harvesting wireless sensor networks," *Lecture Notes in Comp. Sc.: Stabil., Safety, and Secur. of Distr. Syst., 2010,* vol. 6366, pp. 236-251, 2010.

[75] C. Alippi, G. Anastasi, M. Di Francesco and M. Roveri, "Energy management in wireless sensor networks with energy-hungry sensors," *IEEE Instr., & Meas. Mag.,* vol. 12, no. 2, pp. 16-23, 2009.

[76] C.M. Vigorito, D. Ganesan and A.G. Barto, "Adaptive control of duty cycling in energy-harvesting wireless sensor networks," in *IEEE 4th Annual Comm. Soc. Conf. Sensor, Mesh and Ad Hoc Comm. and Netw.,* pp. 21-30, Jun. 2007.

[77] H. Besbes, G. Smart, D. Buranapanichkit, C. Kloukinas and Y. Andreopoulos, "Analytic Conditions for Energy Neutrality in Uniformly-formed Wireless Sensor Networks," *Submitted to IEEE Transactions on Wireless Communications*, - Under review.

[78] L. Benini, A. Bogliolo and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Trans. On Very Large Scale Integr. (VLSI) Syst.,* vol. 8, no. 3, pp. 299-316, 2000.

[79] N. Pettis, L. Cai and Y.-H. Lu, "Dynamic power management for streaming data," *Proc. of the Inter. Symp. on Low Power Elec. and Design, 2004. ISLPED '04.* pp. 62-65, Aug. 2004.

[80] L. Cai and Y.-H. Lu, "Energy management using buffer memory for streaming data," *IEEE Trans. on Comp.-Aided Design of Integ. Circ. and Syst.,* vol. 24, pp. 141-152, Feb. 2005.

[81] C.M. Lien, S.H. Chang, C.S. Chang, and D.S. Lee, "Anchored desynchronization," *Proceedings IEEE INFOCOM*, pp. 2966-2970, 2012.

[82] D. De Guglielmo, G. Anastasi, and M. Conti, "A localized de-synchronization algorithm for periodic data reporting in IEEE 802.15. 4 WSNs", *IEEE Symposium on Computers and Communications (ISCC), IEEE*, pp. 605-610, 2012.

[83] T. Settawatcharawanit, S. Choochaisri, C. Intanagonwiwat, and K. Rojviboonchai, "V-desync: Desynchronization for beacon broadcasting on vehicular networks," *IEEE Vehicular Technology Conference (VTC Spring), IEEE*, pp. 1-5, 2012.

[84] S. Choochaisri, K. Apicharttrisorn, K. Korprasertthaworn, P. Taechalertpaisarn, and C. Intanagonwiwat, "Desynchronization with an artificial force field for wireless networks," *ACM SIGCOMM Computer Communication Review, ACM*, vol. 42, no. 2, pp. 7-15, 2012.

[85] J. Klinglmayr and C. Bettstetter, "Self-organizing synchronization with inhibitory-couples oscillaotrs: convergence and robustness," *ACM Transactions on Autonomous and Adaptive Systems (TAAS),* vol. 7, no. 3, Sept. 2012.

[86] R. Pagliari and A. Scaglione, "Scalable network synchronization with pulse-coupled oscillators," *IEEE Trans. on Mobile Computing,* vol. 10, no. 3, pp. 392–405, 2011.

[87] Y.-W. Peter Hong, A. Scaglione, and R. Pagliari, "Pulse coupled oscillators' primitive for low complexity scheduling," *IEEE International Conference Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2753–2756, 2009.

[88] A. Mutazono, M. Sugano, and M. Murata, "Energy efficient self-organizing control for wireless sensor networks inspired by calling behavior of frogs," *Computer Communications, Elsevier,* 2011.

[89] S. Ashkiani and A Scaglione, "Discrete Dithered Desynchronization," *arXiv preprint arXiv:1210.2122,* 2012.

[90] H. Yamamoto, N. Wakamiya, and M. Murata, "An Inter-Networking Mechanism with Stepwise Synchronization for Wireless Sensor Networks," *Sensors, Molecular Diversity Preservation International,* vol. 11, no. 9, pp. 8241-8260, 2011

[91] T. Nakano, "Biologically inspired network systems: a review and future prospects," *IEEE Trans. on Syst., Man, and Cybernetics, Part C: Applications and Reviews, IEEE,* vol. 41, no 5, pp. 630-343, 2011.

[92] I. Bojic, and V. Podobnik, and I. Ljubi, and G. Jezic, and M. Kusek, "A self-optimizing mobile network: Auto-tuning the network with firefly-synchronized agents," *Information Sciences, Elsevier,* vol. 182, no. 1, pp. 77-92, 2012.

[93] A. Cornejo, and F. Kuhn, "Deploying wireless networks with beeps," *Distributed Computing, Springer*, pp. 148-162, 2010.

[94] G. Strang, and T. Nguyen, *Wavelets and filter banks*, Cambridge University Press, 1996.

[95] A. Sinha, and A. P. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design & Test of Computers*, vol. 18, no. 2,pp. 62-74, 2001.

[96] N. A. Pantazis, D. J. Vergados, D. D. Vergados and C. Douligeris, "Energy efficiency in wireless sensor networks using sleep mode TDMA scheduling," *Ad Hoc Networks ,Elsevier*, vol. 7, no. 2, pp. 322- 343, Mar 2009.

[97] F. Salvadori, M. Campos, P. S. Sausen, R. F. Camargo, C. Gehrke, C. Reach, M. A. Spohn, and A. C. Oliveira, "Monitoring in industrial systems using wireless sensor network with dynamic power management," *IEEE Trans. On Instrumentation And Measurement* ,vol. 58, no. 9, pp. 3104-3111, Sep 2009.

[98] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung, "MAC essentials for wireless sensor networks," *IEEE Comm. Surv. Tut.,* vol. 12, no. 2, pp. 222-248, quarter 2010.

[99] G. WernerAllen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," *7th symposium on Operating systems design and implementation. ACM*, pp. 381-396, 2006.

[100] D. Palma, L. Bencini, G. colloid, G. Manes, F. Chiti, R. Fantacci, and A. Manes, "Distributed monitoring systems for agriculture based on wireless sensor network technology," *International Journal on Advances in Networks and Services*, vol. 3, 2010.

[101] D. Lucarelli, and I-J. Wang, "Decentralized synchronization protocols with nearest neighbor communication," *SenSys '04 Proceedings of the 2nd international conference on Embedded networked sensor systems, ACM*, pp. 62-68, 2004

[102] A. Tyrrell, G. Auer, and C. Bettstetter, "Biologically inspired synchronization for wireless networks," *Series: Studies in Computational Intelligence, Springer*, 2007.

[103] D. Buranapanichkit, A. Vittorioso, G. Fortino and Y. Andreopoulos, "Performance Comparison of Centralized and Distributed Coordination for TDMA Operation in Wireless Sensor Networks," *London Communication Symposium (LCS),* 2011.

[104] A. Vittorioso, D. Buranapanichkit, G. Fortino and Y. Andreopoulos, "Coordination For TDMA Operation In WSNs: Comparison Between Centralized and Distributed Mechanisms," *the 9th European Conference on Wireless Sensor Networks (EWSN)*, 2012

[105] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed randomized TDMA scheduling for wireless ad-hoc networks," *IEEE Trans. Mob. Comput.,* Oct. 2009.

[106] A. Mutazono, M. Sugano, and M. Murata, "Frog call-inspired self-organizing anti-phase synchronization for wireless sensor networks," *Nonlinear Dynamics and Synchronization, INDS IEEE*, pp. 81 – 88, Jul. 2009.

[107] P. Taechalertpaisarn, S. Choochaisri, and C. Intanagonwiwat, "An orthodontics-inspired desynchronization algorithm for wireless sensor networks," *Communication Technology (ICCT), 2011 IEEE*, pp. 631-636, Sept. 2011.

[108] H. Kang and J. L. Wong, "A localized multi-hop desynchronization algorithm for wireless sensor networks," *INFOCOM 2009, IEEE*, pp. 2906-2910, Apr. 2009.

[109] C. Mühlberger, R. Kolla, "Extended desynchronization for multi-hop topologies," *Technical Report, Institut für Informatik, Universität Würzburg*, 2009. http://www5.informatik.uniwuerzburg.de/publications/techreports/tr_extdesync.pdf

[110] R. Mirollo and S. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM Journal of Applied Mathematics*, vol. 50, no. 6, pp. 1645-1662, 1990.

[111] S. H. Strogatz and I. Stewart, "Coupled oscillators and biological synchronization," *Scientific American,* 1993

[112] Y. W. Hong and A. Scaglione, "Time synchronization and reach-back communications with pulse-coupled oscillators for UWB wireless ad hoc networks," *Ultra Wideband Systems and Technologies, 2003 IEEE*, pp. 190-194, Nov. 2003.

[113] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 5, no. 5, Feb. 2009.

[114] A. Ghosh and S. K. Das, "Coverage and connectivity issues in wireless sensor networks: a survey," *Pervasive and Mobile Computing, Elsevier*, vol. 4, no. 3, pp. 303-334, Jun. 2008.

[115] A. S. Hu and S. D. Servetto, "On the scalability of cooperative time synchronization in pulse-connected networks," *IEEE Trans. Information Theory*, vol. 52, no. 6, pp. 2725-2748, Jun. 2006.

[116] R. Mangharam, A. Rowe, and R. Rajkumar, "Firefly: A cross-layer platform for real-time embedded sensor networks," *Real Time Systems Journal, Springer*, vol. 37, no. 3, pp. 183-231, Dec. 2007.

[117] X. Y. Wang and A. B. Apsel, "Pulse coupled oscillator synchronization for low power UWB wireless transceivers," *Circuits and Systems, MWSCAS 2007*, pp. 1524-1527, Aug. 2007.

[118] E. Mallada and A. Tang, "Synchronization of phase-coupled oscillators with arbitrary topology," *American Control Conference (ACC), 2010*, pp. 1777-1782, Jul. 2010.

[119] S. R. Campbell, D. L. Wang, and C. Jayarakash, "Synchrony and desynchrony in integrate-and-fire oscillators," *Neural Comput.*, vol. 11, no. 7, pp. 1595-1619, Oct. 1999.

[120] M. B. H. Rhouma and H. Frigui, "Self-organization of pulse-coupled oscillators with applications to clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 2, pp. 180-195, Feb. 2001.

[121] R. Mathar and J. Mattfeldt, "Pulse-coupled decentral synchronization," *SIAM J. Appl. Math.*, vol. 56, no. 4, pp. 1067-1116, Aug. 1996.

[122] A. Dailot, D. Dolev, and H. Parnas, "Self-stabilizing pulse synchronization inspired by biological pacemaker networks," *Lecture Notes in Compu. Science: Self-Stabilizing Systems, Springer*, vol. 2704, pp. 32-48, 2003.

[123] http://ubi.cs.washington.edu/files/imote2/docs/PXA27x_Developers_Manual-280000003.pdf

[124] D. J. Huang, K. J. You, and W. C. Teng, "Secured flooding time synchronization protocol," *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE*, pp. 620-625, Oct. 2011.

[125] U. Schmid and K. Schossmaier, "Interval-based clock synchronization," *Real-Time Systems, Springer*, vol. 12, no. 2, pp. 173-228, Mar. 1997.

[126] P. Blum, L. Meier, and L. Thiele, "Improved interval-based clock synchronization in sensor networks," *Information Processing in Sensor Networks, 2004. IPSN*, pp. 349-358, Apr. 2004.

[127] J.F. Mingorance-Puga, G. Maciá-Fernández, A. Grilo and N. M. C. Tiglao, "Efficient multimedia transmission in wireless sensor networks," *Next Generation Internet (NGI), 2010 6th EURO-NF Conference on IEEE*, pp. 1-8, Jun. 2010.

[128] L. Shu, M. Hauswirth, L. Wang, Y. Zhang and J. H. Park, "Cross-layer optimized data gathering in wireless multimedia sensor networks," *Computational Science and Engineering, CSE'09 International Conference on IEEE*, vol. 2, pp. 961-966, 2009.

[129] K. T. Phan, R. Fan, H. Jiang, S. A. Vorobyov and C. Tellambura, "Network lifetime maximization with node admission in wireless multimedia sensor networks," *IEEE Trans. on Vehicular Technology*, vol. 58, no. 7, pp. 3640-3646, 2009.

[130] D. Kandris, M. Tsagkaropoulos, I. Politis, A. Tzes and S. Kotsopoulos, "Energy efficient and perceived qos aware video routing over wireless multimedia sensor networks," *Ad Hoc Networks, Elsevier*, vol. 9, no. 4, pp. 591-607, Jun. 2011.

[131] A. Newell and K. Akkaya, "Self-actuation of camera sensors for redundant data elimination in wireless multimedia sensor networks," *Communications, 2009. ICC '09. IEEE International Conference*, pp. 1-5, Jun. 2009.

[132] W. Yu, Z. Sahinoglu and A. Vetro, "Energy efficient jpeg 2000 image transmission over wireless sensor networks," *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, vol. 5, pp. 2738-2743, Dec. 2004.

[133] A. Redondi, M. Cesana and M. Tagliasacchi, "Rate-accuracy optimization in visual wireless sensor networks," *19th IEEE International Conference on Image Processing (ICIP)*, pp. 1105-1108, Oct. 2012.

[134] R. Adler, M. Flanigan, J. Huang, R. Kling, N. Kushalnagar, L. Nachman, C.-Y. Wan, and M. Yarvis, "Intel mote 2: an advanced platform for demanding sensor network applications," *the 3rd International Conference on Embedded Networked Sensor Systems*, vol. 2, no. 4, pp. 298–298, 2005.

[135] P. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, J. D. Tygar, and S. Sastry, "Citric: A low-bandwidth wireless camera network platform," *Second ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC 2008*, pp. 1-10, Sept. 2008.

[136] S. Paniga, L. Borsani, A. Redondi, M. Tagliasacchi and M. Cesana, "Experimental evaluation of a video streaming system for wireless multimedia sensor networks," *Ad Hoc Networking Workshop (Med-Hoc-Net), 2011 The 10th IFIP Annual Mediterranean*, pp. 165-170, Jun. 2011.

[137] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, Nov. 2004.

[138] E. Rosten, R. Porter and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010. [Online]. Available: http://lanl.arXiv.org/pdf/0810.2434

[139] M. Calonder, V. Lepetit, C. Strecha and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[140] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister "OpenWSN: a standards-based low-power wireless development environment." *Europ. Trans. on Emerg. Telecom. Technol*., vol. 23, no. 5, pp. 480-493, Aug. 2012.

[141] IEEE Standard for Local and Metropolitan Area Networks, Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer, *IEEE Computer Society*, April. 16, 2012, online at: http://standards.ieee.org/getieee802/Download/802.15.4e-2012.pdf

[142] J. Chen, S. Sheu, and C. Yang, "A New Multichannel Access Protocol for IEEE 802.11 Ad Hoc Wireless LANs," *Proc. 14th IEEE Int'l Symp. Personal, Indoor and Mobile Radio Comm. (PIMRC '03)*, vol. 3, pp. 2291-2296, Sept. 2003.

[143] http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page

[144] http://www.contiki-os.org/index.html

[145] http://www.eecs.harvard.edu/ssc/sync

# C. Supplementary Materials

```
Event FiringMessage.Received{
   set slot_no
   keep AllReceived_FireTime
}
Event Time.AlarmFired{
   if (just_usedAlgTime == FALSE)
      call CheckEvent()
      dataTime_start = AlgorithmTime
      dataTime_stop = Next_FireTime
      call SetPhysicalTime()
   else
      call SendFireMessage()
      call SetAlgorithmTime()
}
Task SetPhysicalTime{
   slot_duration = (dataTime_stop - dataTime_start)/total_nodes
   slot_start = slot_duration * (slot_no-1)
   slotTime_MyNode = dataTime_start + slot_start
   slotTime_EndNode = slotTime_MyNode + slot_duration
   call SetAlarmTime(slotTime_MyNode)
   just_usedAlgTime  = TRUE
}
Task SetAlgorithmTime{
   call SetAlarmTime(T)
   just_usedAlgTime  = FALSE
}
```

Figure C.1: Code of SYNC algorithm

```
Task CheckEvent{
   sort AllReceived_FireTime
   phase =  (Next_FireTime - Received_FireTime)
   if (phase < refractory_period || phase > (T - refractory_period))
      just_Phase0 = TRUE
      return
   else
      just_Phase0 = FALSE
      if (phase >= T/2)                                          // Phase A
         phase = phase + epsilon
         if (phase >= T)                                         // Phase I.A
            Next_FireTime = Received_FireTime + rand_Number*Duration + Offset
         else                                                    // Phase II.A
            Next_FireTime = Next_FireTime + epsilon
      else                                                       // Phase B
         phase = phase - epsilon
         if (phase <= 0)                                         // Phase I.B
            Next_FireTime = Received_FireTime + rand_Number*Duration + Offset
         else                                                    // Phase II.B
            Next_FireTime = Next_FireTime - epsilon
}
```

Figure C.2: Code of SYNC algorithm when considering the phase.

```
Event FireMessage.Received{
   if (Rx_mode == TRUE)
      call FlashWrite()
      Rx_mode = FALSE
}
Event DataMessage.Received{
   get DataMessage
   Rx_mode = TRUE
}
Event FireMessage.SendDone{
   if (number_nodes == no_neighbors && just_Phase0 == TRUE)
      set ReceiverID
      call DataOut()
}
Event DataMessage.SendDone(
   get CurrentTime
   if ((NextNode_FireTime + T) - CurrentTime < time_SendData_onePack)
      call FlashCircRead()
   else
      call DataOut()
)
Task DataOut{
   call FlashRead()
   call SendDataMessage()
}
```

Figure C.3: Code of SYNC algorithm when transmitting data.

```
Event Time.AlarmFired{
   Current_FireTime = get CurrentTime
   call SendFireMessage()
   just_Fired = TRUE
   if (prev_Received == TRUE)
      call SetAlarmTime(T)
}
Event FireMessage.Received{
   get CurrentTime
   if (just_Fired == TRUE)
      just_Fired = FALSE
      prev_Received = FALSE
      NextNode_FireTime = CurrentTime
      call CalculatedNextFireTime()
   else
      LastNode_FireTime = CurrentTime
}
Task CalculatedNextFireTime{
   Next_FireTime = T + (1-alpha)* Current_FireTime
                  + (alpha)*((NextNode_FireTime + LastNode_FireTime)/2)
   call SetAlarmTime(Next_FireTime)
}
```

Figure C.4: Code for DESYNC-TDMA.

```
Event FireMessage.Received{
   if (Rx_mode == TRUE )
      call FlashWrite()
      Rx_mode = FALSE
}
Event DataMessage.Received{
   get DataMessage
   Rx_mode = TRUE
}
Event FireMessage.SendDone{
   if (set_converged == TRUE)
      set ReceiverID
      call DataOut()
}
Event DataMessage.SendDone(
   get CurrentTime
   if ((FireTime_NextNode + T) - CurrentTime < time_SendData_onePack)
      call FlashCircRead()
   else
      call DataOut()
)
Task CalculatedNextFireTime(){
   if (abs((Next_FireTime - Current_FireTime) - T) < threshold)
      set_converged = TRUE
   else
      set_converged = FALSE
}
Task DataOut{
   call FlashRead()
   call SendDataMessage()
}
```

Figure C.5: Code for Desync-TDMA when transceiving data.

```
Event FireMessage.Received{
   Received_CurrentTime = get CurrentTime
   diff_time = Received_CurrentTime - Current_FireTime
   Non_listen_phase =  T * (1-(1/numberNodes))
   if(diff_Time > Non_listen_phase && diff_Time < T)
      Next_FireTime = Received_CurrentTime + rand_Number*Duration + Offset
      call SetAlarmTime(Next_FireTime)
      listen_phase = TRUE
   else
      listen_phase = FALSE
}
Event Time.AlarmFired{
   Current_FireTime = get CurrentTime
   if (listen_phase == FALSE)
     call SendFireMessage()
     call SetAlarmTime(T)
   else
     call setAlarmFired()
}
Task setAlarmFired{
   delay_fire = T - ((1-alpha)*diff_time + alpha * Non_listen_phase)
   Next_FireTime = Received_CurrentTime + delay_fire
   call SetAlarmTime(Next_FireTime)
   listen_phase = FALSE
   if (abs((Next_FireTime - Current_FireTime) - T) < threshold)
      set_converged = TRUE
   else
      set_converged = FALSE
}
```

Figure C.6: Code of PCO-based inhibitory coupling.

We also provide a TinyOS nesC implementation of the proposed distributed TFDMA online [45].

### C.7 Distributed TFDMA Pseudocode

***Events:***

```
1   init(){
2      // configurable system parameters
3      T = 0.25 sec; alpha = 0.95; P_sw = 0.33; beta = 1.25; Z = 60; C = 3;
4      TDMA_threshold = 20 ms;          // TDMA convergence threshold
5      channel_to_join = c/rand; s_d = 1; // all nodes join in a random Ch;set switch direction
6      // state flags
7      just_fired = False; TDMA = False; switch_mode = False; just_switched = False;
8      // state variables
9      next_fire = last_fire = 0;       // detected Fire times of previous and next node
10     tot_sw_period = tot_Z_period = 0;// period counters during: switch mode & switching inactivity
11     W_prev = W_new = 0;              // total nodes in previous & new channel (for node switching)
12     my_scheduled_fire = Now + T;     // time that the node is scheduled to broadcast own Fire msg
13     // initialization here
14     call SwitchChannel(channel_to_join); call SetFireTimer(my_scheduled_fire);
15  }
16
17  on_Fire_timer_expire(){             // event to handle node's own Fire msg broadcast
18     if (just_switched == False){
19        call SendFireMsg();
20        if (switch_mode == False){ call CheckForSwitching(); } // allows switching during convergence
21        just_fired = True; W_prev = 0;
22     }else{ call CalculateChannelToStay(); }
23  }
24
25  on_receive_Fire_Msg(){              // event to handle reception of Fire msg
26     if (just_fired == True){         // this msg came just after own Fire msg-->calc next Fire time
27        just_fired = False; next_fire = Now; call CalculateNextFireTime();
28     }else{ last_fire = Now; }        // this msg becomes the last Fire msg we heard before our own
29     if (just_switched == False){ W_prev++; }else{ W_new++; }
30  }
31
32  on_receive_Switch_Msg(type){        // event to handle reception of Switch msg
33     if (type == RETURN) {
34        P_sw = P_sw/beta; s_d = -s_d; // node returns --> decrease P_sw, change switch direction
35        switch_mode = False; just_switched = False; W_prev = W_new = 0;
36     }else{ switch_mode = True; just_switched = False; }
37  }
```

***Supporting Tasks:***

```
38  Task CalculateNextFireTime(){      // calculates own scheduled Fire time via reactive listening
39     if (switch_mode == True){
40        my_scheduled_fire = Now + T;  // in switch mode we fire every T sec
41        if (just_switched == False){  // for nodes remaining in the same channel
42           tot_sw_period++;
43           if (tot_sw_period == 2){   // no switch msg received --> increase P_sw
44              P_sw = P_sw*beta; tot_sw_period = 0;
45           }
46        }
47     }else{ my_scheduled_fire = T + (1-alpha)* Now + (alpha)*((next_fire + last_fire)/2); }
48     call SetFireTimer(my_scheduled_fire);
49     if (my_scheduled_fire - Now – T < TDMA_threshold){ TDMA = True; }else{ TDMA = False; }
50  }
51
52  Task CalculateChannelToStay(){     // decide on channel switching based on reactive listening
53        if ((W_new + 2) > W_prev){
54           call SendSwitchMsg(RETURN);// return to old Ch and broadcast RETURN msg
55           s_d = -s_d;               // change switch direction (for next time)
56        }                            // else stay in the same channel
57        switch_mode = False; just_switched = False; W_prev = W_new = 0;
58  }
59
60  Task CheckForSwitching(){          // performs channel switching (via P_sw or after Z periods)
```

```
61      rand_num = rand();              // random number between (0,1)
62      if (rand_num <= P_sw OR tot_Z_period == Z){
63         call SendSwitchMsg(SWITCH)    // broadcast SWITCH msg in old Ch before switching to new Ch
64         channel_to_join += s_d;       // switch to the next or previous channel (since s_d = +/-1)
65         if (channel_to_join > C) channel_to_join=1;
66         if (Channel_to_join < 0) channel_to_join=C;
67         W_new = 0; call SwitchChannel(channel_to_join);
68         P_sw = P_sw/beta; just_switched = True; tot_Z_period = 0;
69      }else{ tot_Z_period++; }
70   }
```

**Notes on Pseudocode C.7:**

- Configurable system parameters: For details on the utilized parameters, please see in chapter 3. All nodes are set by default to join channel with a random number between 1 and C or one of the $C$ available channels randomly (`channel_to_join` in line 5). The provided NesC code leads to balanced TFDMA for arbitrary sets of parameters; as such the system parameters per node can be configured according to the throughput/delay constraints and the connectivity constraints of a particular application and they don't need to be identical in all sensor nodes.

- Explanation of State flags:
  - `just_fired`: This flag becomes `True` once the node broadcasts its own Fire message (line 21). It is used to identify the first Fire message received *after* the node's own Fire message (lines 26-27).
  - TDMA: This flag becomes `True` once desynchronization to TDMA within the current channel has been reached (line 49).
  - `switch_mode`: This flag becomes `True` once a switch message is received with the SWITCH argument (line 36). It controls: *(i)* if switching can be pursued by the node or not (line 20); *(ii)* whether the node's own Fire message broadcast will be repeated every Ts (lines 39, 40), or if it will be set via reactive listening (line 47).
  - `just_switched`: This flag becomes `True` once the node switches a channel (line 68) to listen to Fire messages for channel switching based on reactive listening. It controls: *(i)* whether to send a Fire message, or simply check for Fire messages received in order to decide on which channel to remain to (line 18, `CalculateChannelToStay()` in line 22); *(ii)* whether the node will increment `W_prev` or `W_next` (line 29); *(iii)* whether the node will increment the number of periods it will wait for switch mode to expire (line 41,42), which will happen if `switch_mode = True` and `just_switched = False`.

- General remarks:
  - We do not explicitly present the handling of the case when a node joins a channel as the only node, in which case it will have to fire independently every Ts. This requires trivial modifications and it is already supported in the provided NesC source code.
  - The provided pseudocode does not explicitly take into account whether the new channel that the node attempts to switch to is in switch mode as well (footnote 3 of the chapter 3). This is however taken into account in the provided source code.
  - The provided pseudocode and the NesC implementation support only the case of $W_{\mathrm{sw}} = 1$ that has been proven to be stable via Proposition 1 of the chapter 3.
  - In the pseudocode presented, s_d alternates between +1 and -1 (lines 34 & 55). For C>3, this is enhanced by gradually increasing s_d via the use of the values:
    - `[+1, -1, +2, -2, +3, -3, +4, -4, +5, -5, +6, -6, +7, -7, +8, –8]`

The limit is $\pm 8$ since IEEE802.15.4 supports only up to 16 channels. In general, $s_c$ is increased up to $\pm\lfloor C/2 \rfloor$ to cover all possible channels, as mentioned before eq. (1) in chapter 5. This is dealt with in a straightforward manner in the provided NesC code via the use of a vector with the above values; we do not include this in the pseudocode for brevity of description.

- The provided implementation handles all events via a single timer, since only the fire messages need to be scheduled and all other decisions are reactive based on the messages received. This follows the PCO TDMA approaches. This is an important aspect that demonstrates the practicality of the proposed TFDMA approach: time-frequency balancing can be achieved without requiring an additional timer or any form of complicated hardware support.

**C.2 Implementation the iMote2 for TinyOS 1.x**

% TinyOS iMote2 NesC code for desynchronization
% This code produces the multi-channel desynchronization of the manuscript:
% D. Buranapanichkit and Y. Andreopoulos, "Distributed Time-Frequency Division Multiple
% Access Protocol For Wireless Sensor Networks", *IEEE Wireless Communications Letters*,
% submitted.
% The provided code in folder `/DesyncSwitch` is including only the basic functionality of
% desynchronization. This was done for ease of illustration of the algorithms tested.
% The code is set for 4 channels; the total number channels can be easily selected within the source code.

**Installing the iMote2 Development Environment for TinyOS 1.x**

*step 1* Installing Cygwin if you use Windows
- Download and install the latest version of Cygwin from Sourceware (http://sourceware.org/cygwin/)

*step 2* Downloading the TinyOS1.x source code
- Install the latest version of TinyOS 1.x from SourceForge
- Configuring the TinyOS1.x Tree for iMote2, see http://shm.cs.uiuc.edu/files/GettingStarted_Imote2.pdf (this document has complete instructions for all the steps required before you can use the provided code for desynchronization)

*step 3* Installing the NesC compiler
- Download the latest version of the compiler from SourceForge

*step 4* Installing the Wasabi tool suite
- Download the wasabi tool suite from the Intel website (http://www.intel.com/design/intelxscale/dev_tools/031121/ )

**Testing the application (instructions refer to Windows and Cygwin, straightforward modifications are required for usage under Linux)**

*step 1* Create the new folder for the application at directory `c:\tinyos\cygwin\opt\apps`
*step 2* Update the interface file of the TinyOS 1.x for the iMote2
- CC2420RadioM.nc file at directory `c:\tinyos\cygwin\opt\tos\lib\CC2420Radio`

For the number of Imotes you would like to test with, repeat the steps below before using them to measure.

*step 3* Compile the application for the iMote2 in the Cygwin shell:

```
cd $TOSROOT/apps/Desync
ADDRESS=x make imote2
```

with x being the node id

*step 4* Plug the USB cable to the PC and directly to the Crossbow iMote2 (http://www.xbow.com). Turn Imote2 on.

*step 5* Download the iMote2 application with the USB loader:

```
USBLoaderHost –p build/imote2/main.bin.out
```

Once all motes have been loaded with the application, you can also configure another 4 Imotes as base stations to record all messages in the 4 channels used. Switching on the base stations and the test motes produces the results of the manuscript.