M.A. Labrador
P.M. Wightman

Topology Control in Wireless Sensor Networks

With a Companion Simulation Tool for Teaching and Research

Springer

Topology Control in Wireless Sensor Networks

Miguel A. Labrador · Pedro M. Wightman

Topology Control in Wireless Sensor Networks

– with a companion simulation tool for teaching and research –

Springer

Miguel A. Labrador
University of South Florida
Dept. Computer Science & Engineering
4202 E. Fowler Ave.
Tampa, FL 33620
USA
labrador@cse.usf.edu

Pedro M. Wightman
University of South Florida
Dept. Computer Science & Engineering
4202 E. Fowler Ave.
Tampa, FL 33620
USA
pedrow@cse.usf.edu

ISBN 978-1-4020-9584-9 e-ISBN 978-1-4020-9585-6

Library of Congress Control Number: 2008941077

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Dedicado a mi esposa Mariela, y a mis hijos Miguel Andrés y Daniel Ignacio, mis más grandes tesoros.

Miguel A. Labrador

To my three girls: Astrid, Hillary and Sophia; my foundation, my support, my inspiration.

Pedro M. Wightman

Preface

The field of wireless sensor networks continues to evolve and grow in both practical and research domains. More and more wireless sensor networks are being used to gather information in real life applications. It is common to see how this technology is being applied in irrigation systems, intelligent buildings, bridges, security mechanisms, military operations, transportation-related applications, etc. At the same time, new developments in hardware, software, and communication technologies are expanding these possibilities. As in any other technology, research brings new developments and refinements and continuous improvements of current approaches that push the technology even further.

Looking toward the future, the technology seems even more promising in two directions. First, a few years from now more powerful wireless sensor devices will be available, and wireless sensor networks will have applicability in an endless number of scenarios, as they will be able to handle traffic loads not possible today, make more computations, store more data, and live longer because of better energy sources. Second, a few years from now, the opposite scenario might also be possible. The availability of very constrained, nanotechnology-made wireless sensor devices will bring a whole new world of applications, as they will be able to operate in environments and places unimaginable today. These two scenarios, at the same time, will both bring new research challenges that are always welcome to researchers.

Book Origin and Overview

This book is the result of more than six years of research in wireless sensor networks. This research involved investigating new techniques for localization and localization services, energy-efficient MAC, network, and transport layer protocols, and more recently, topology control, the main topic of the book. Not surprisingly, although the book emphasizes topology control, it also includes background information on communication protocols for wireless sensor networks.

The book is divided in three parts. Part I consists of six chapters containing general information about wireless sensor networks, communication protocols, and topology control. Chapter 1 is an introductory chapter that describes the architec-

ture of a generic wireless sensor device and network architectures. The chapter ends with the motivation for these types of networks describing the possible application domains and the challenges that still remain. Chapter 2 describes the most important aspects of the physical layer as they relate to wireless sensor networks. This is not a typical chapter on physical layer communication technologies; instead, it includes needed information about signal propagation models, energy dissipation models, error generation models in wireless networks, and sensing models, all of them of utmost importance in the design and evaluation of wireless sensor networks. Chapter 3 is about the Data Link Layer of the communication protocol stack, and as such, includes the Medium Access Control protocols for energy efficient access of the wireless media and Logical Link Control protocols for flow and error control. The topic of routing for wireless sensor networks is included in Chapter 4 where the most important routing protocols are surveyed and explained. Chapter 5 is devoted to transport layer protocols for wireless sensor networks. Protocols for applications with different reliability requirements are explained along with a discussion about the need of congestion control and the use of TCP and UDP in wireless sensor networks. Finally, Chapter 6 introduces the reader to the topic of topology control and provides the road map for the rest of the book. In this chapter, the reader is presented with the motivations for topology control, its challenges, and general design guidelines. A formal definition of topology control is presented along with a discussion about where in the communication protocol stack this function should be implemented. Lastly, a new taxonomy of topology control is presented that includes the concept of topology construction (currently known as topology control), and for the first time, the concept of topology maintenance.

Part II of the book is devoted to what we call Topology Construction, or techniques that, given a set of nodes, build a reduced topology to save energy while preserving important network characteristics, such as network coverage and connectivity. Chapter 7 discusses those topology construction mechanisms that build the reduced topology by controlling the transmission power of the nodes. Chapter 8 includes those techniques that build reduced hierarchical topologies. The last chapter of Part II is Chapter 9, which includes hybrid topology construction techniques for the first time.

The third and last part of the book, is about Topology Maintenance, a concept that had never been formally defined as part of topology control. Chapter 10 introduces the topic and includes general information about topology maintenance that applies to all three remaining chapters, such as design issues, topology maintenance triggering criteria, and radio synchronization. Chapter 11 introduces topology maintenance static techniques and includes a performance evaluation of global static techniques in sparse and dense networks. Chapter 12 is about topology maintenance dynamic techniques, and it also includes a performance evaluation of both global and local dynamic techniques in sparse and dense networks. Finally, Chapter 13, which ends Part III of the book, includes topology maintenance hybrid techniques and their performance evaluation, and two sections where all topology maintenance techniques described in this part of the book are further evaluated and compared.

Intended Audience

The book is intended for graduate students, professors, researchers, and industry professionals interested in wireless sensor networks. The book can be used as a reference book in a graduate class on wireless sensor networks, or as the main book in an advanced, research oriented course on the same topic with emphasis on topology control. The Atarraya simulator is an excellent teaching aid for explaining difficult concepts in a graphical way, and an invaluable tool for experimentation and the assignment of research-oriented projects for the class.

Resources

Appendix A is another important contribution of this book. It describes the structure of the Atarraya simulator in detail. Atarraya is a Java-based simulation tool developed for teaching and researching topology control topics in wireless sensor networks. We hope that you use the tool as much as we have in your classes and in your research. As a Java-based tool, it is easily understandable and expandable, so you can include new topology control mechanisms as well as other aspects of wireless sensor networks, such as new propagation models, error models, etc. Atarraya comes with a graphical user interface that can be used to visualize the effect of applying topology control algorithms in a class or demonstration. All experimental-based figures included in the book were generated with Atarraya.

Atarraya, which is copyrighted under the GNU license agreement, can be downloaded for free from *http://www.csee.usf.edu/~labrador/Atarraya*.

Acknowledgments

We are very grateful to Hillary Wade, who spent a tremendous amount of time revising and making suggestions about the manuscript. We would also like to thank the staff of Springer for their support during all the phases of the book. Finally, we want to acknowledge our own families for their patience, support, and understanding during all these months of continuous, hard work.

About the Authors

Miguel A. Labrador received the M.S. in Telecommunications and the Ph.D. degree in Information Science with concentration in Telecommunications from the University of Pittsburgh, in 1994 and 2000, respectively. Since 2001, he has been with the University of South Florida, Tampa, where he is currently an Associate Professor in the Department of Computer Science and Engineering. Before joining USF, he worked at Telcordia Technologies, Inc., NJ, in the Broadband Networking Group of the Professional Services Business Unit. He has more than fifteen years of industry experience in the Telecommunications area. His research interests are in design

and performance evaluation of computer networks and communication protocols for wired, wireless, and optical networks, energy-efficient mechanisms for wireless sensor networks, bandwidth estimation techniques, and location-based services. He has published more than 50 technical and educational papers in journals and conferences devoted to these topics. Dr. Labrador has served as Technical Program Committee member of many IEEE conferences and is currently member of the Editorial Board of Computer Communications and the Journal of Network and Computer Applications, Elsevier Science. He served as Secretary of the IEEE Technical Committee on Computer Communications (TCCC), Chair of the IEEE VTC 2003 Transport Layer Protocols over Wireless Networks Symposium, and guest editor of the special issue of Computer Communications on "Advanced Location-Based Services". Dr. Labrador is a senior member of the IEEE Communications Society, and member of the ACM SIGCOMM, ASEE, and Beta Phi Mu honor society.

Pedro M. Wightman received his B.S. in Systems Engineering from the Universidad del Norte, in Barranquilla, Colombia, in 2004. He received his M.S. in Computer Science from the University of South Florida in 2007, where he is a Ph.D. candidate in the department of Computer Science and Engineering. Pedro worked as a part-time assistant professor at the Universidad del Norte during 2004 and 2005. In 2005 he was selected to participate in the National Program of Young Researchers in Colombia, sponsored by the Colombian Institute of Science and Technology, Colciencias. In 2005, he was selected by the Universidad del Norte to participate in the Teaching Formation Program which gave him the opportunity to enroll in the University of South Florida to pursue his doctorate degree in Computer Science and Engineering. His research interests are in the development of energy efficient protocols for wireless sensor networks, topology construction and topology maintenance protocols. He is also interested in the design and implementation of communication platforms with an emphasis on virtual reality chat rooms and distance learning applications. Mr. Wightman has published technical papers in peer-reviewed conferences related to the areas previously mentioned. He is a member of the IEEE Communication Society, and co-founder of CommNet, the Communication Networks Group at USF.

Tampa,
October 2008

Miguel A. Labrador
Pedro M. Wightman

Contents

Part I
Introduction to Wireless Sensor Networks and Topology Control

Chapter 1
Wireless Sensor Networks

1.1 Introduction

Recent advances in sensor and wireless communication technologies in conjunction with developments in microelectronics have made available a new type of communication network made of battery-powered integrated wireless sensor devices. Wireless Sensor Networks (WSNs), as they are named, are self-configured and infrastructureless wireless networks made of small devices equipped with specialized sensors and wireless transceivers. The main goal of a WSN is to collect data from the environment and send it to a reporting site where the data can be observed and analyzed. Wireless sensor devices also respond to queries sent from a "control site" to perform specific instructions or provide sensing samples. Finally, wireless sensor devices can be equipped with actuators to "act" upon certain conditions. These networks are sometimes more specifically referred as Wireless Sensor and Actuator Networks.

At present time, due to economic and technological reasons, most available wireless sensor devices are very constrained in terms of computational, memory, power, and communication capabilities. This is the main reason why most of the research on WSNs has concentrated on the design of energy- and computationally-efficient algorithms and protocols, and the application domain has been restricted to simple data-oriented monitoring and reporting applications. However, all this is changing very rapidly, as WSNs capable of performing more advanced functions and handling multimedia data are being introduced. New network architectures with heterogeneous devices and expected advances in technology are eliminating current limitations and expanding the spectrum of possible applications for WSNs considerably.

This chapter provides a general view of wireless sensor networks describing the node and network architectures, examples of application domains, and the main challenges faced by WSNs with an emphasis on energy conservation.

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

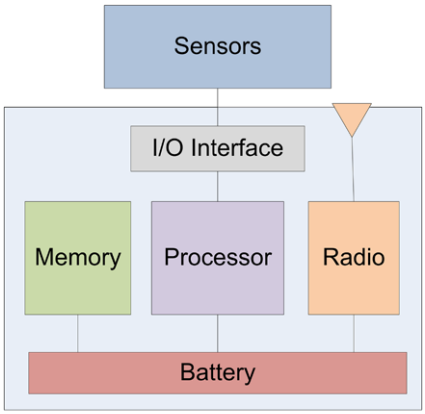


Fig. 1.1 General architecture of a wireless sensor device.

1.2 Node and Network Architectures

1.2.1 Wireless Sensor Device Architecture

The architecture of a wireless sensor device is very simple. As shown in Figure 1.1, a wireless sensor device contains a processor, memory, a radio transceiver and antenna, a power source, and an input/output interface that allows the integration of external sensors into the wireless device. Wireless sensor devices are equipped with low-power and computationally constrained microcontrollers. Microprocessors are not usually included in WSNs because of the cost and because of the simple and specialized tasks that wireless sensor devices are supposed to perform. Also, microcontrollers consume less energy than more advanced CPUs, an important consideration in WSNs since wireless sensor devices are usually battery powered. Further, they are very well suited for WSNs because some parts of the microcontrollers can be set to sleep, if so needed. Normally, microcontrollers for WSNs are either 16- or 32-bit Reduced Instruction Set Computer (RISC) architectures running at low frequencies. Examples of well-known microcontrollers used in WSNs are the Intel StrongARM SA-1100 [57], a fairly high-end processor running at 206 MHz, and the Atmel ATmega 128L [4], an 8-bit microcontroller.

In wireless sensor devices there is also the need for Random Access Memory (RAM) to store application related data such as sensor readings, and Read-Only Memory (ROM) to store program code, which needs to be stored permanently, even if power supply is interrupted. RAM memory is usually included in the microcontroller and is restricted to a few hundreds of kilobytes. The advantage is that the energy needed to drive this memory is usually included in the power consumption figures of the microcontroller. Normally, wireless sensor devices include a few hundreds of Electrically Erasable Read-Only Memory (EEPROM) for configuration data and more memory in the form of FLASH memory, which not only preserves the data in the event of power interruptions but also can be used as RAM memory when needed. FLASH memory is available by external memory devices connected to the

node through USB interfaces. FLASH-based thumb drives and memory sticks provide the memory space needed at lower energy cost than off-chip RAM and ROM memory.

A wireless sensor node also contains a radio transceiver. Radio transceivers contain all necessary circuitry to send and receive data over the wireless media: modulator and demodulator modules, digital to analog and analog to digital converters, low noise and power amplifiers, filters, mixers and antenna are among the most important components. Radio transceivers usually work half-duplex as simultaneous transmission and reception of data over the wireless media is impractical.

Similar to microcontrollers, transceivers can operate in different modes. Normally, transmit, receive, idle and sleep operational modes are available. The sleep mode is a very important energy saving feature in WSNs. However, turning the transceiver on and off must be done carefully, as it may take the transceiver an appreciable amount of energy to turn the circuitry on again. Some transceivers offer additional capabilities, including several sleep modes that turn different parts of the hardware on and off.

Commercially available transceivers for WSNs have different characteristics and capabilities. Normally, they work on three different frequency ranges: 400 MHz, 800–900 MHz, and 2.4 GHz or the Industrial, Scientific, and Medical (ISM) frequency band. Transmission power, modulation schemes, and data transmission rates vary from vendor to vendor. For example, the TR1000 family from RF Monolithics [57] works in the 800–900 MHz range, can dynamically change its transmission power up to 1.4 mW, and transmit up to 115.2 Kbps. The popular Mica motes from Crossbow are equipped with the Chipcom family of transceivers [25]. MICA2 motes have the Chipcom CC100 chip which operates in the 800/900 MHz range, offers up to 50 programmable channels, uses FSK modulation, and has programmable output power. The Chipcom CC1000 draws 27 mA when transmitting at its maximal power, 10 mA when receiving, and less than 1 μA in the sleep mode. The Chipcom CC2420 is included in the MICAZ mote that was built to comply with the IEEE 802.15.4 standard for low data rate and low cost wireless personal area networks [53].

Wireless sensor devices are usually powered by two AA batteries externally attached to the node. Normally, AA batteries store 2.2 to 2.5 Ah at 1.5 V. However, these numbers vary depending on the technology utilized. For example, Zinc–air-based batteries have higher capacity in Joules/cm^3 than lithium batteries. Alkaline batteries have the smallest capacity, normally around 1200 J/cm^3.

Finally, wireless sensors devices are equipped with sensor boards, which contain application-specific sensors. The variety of sensors and sensor boards that can be directly interfaced with the wireless sensor device is very large. Temperature, air quality, pressure, magnetometers, light, acoustic, and accelerometers, are just a small sample of the types of commercially available sensors. This interfacing flexibility is the cause of the wide popularity of WSNs, as they serve as a general platform to solve practical problems in many application domains.

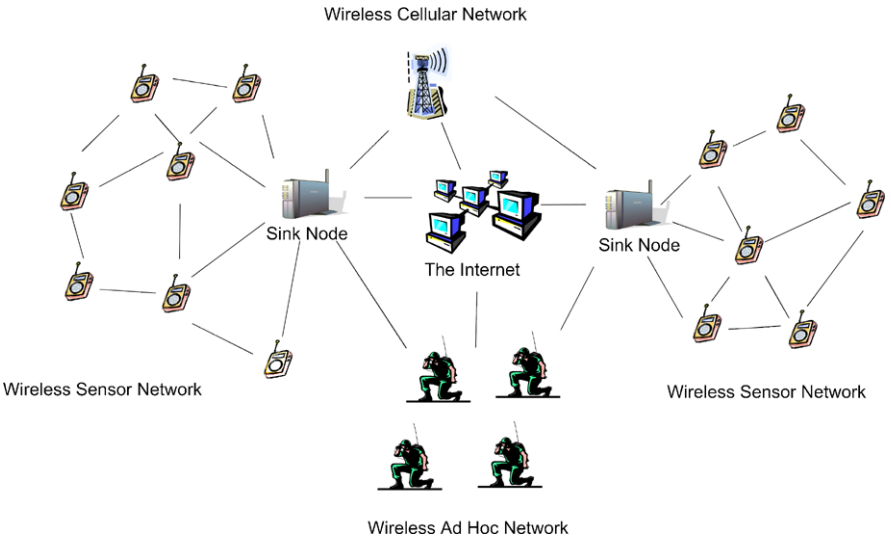


Fig. 1.2 General wireless sensor network architecture.

1.2.2 Network Architectures

Wireless sensor network architectures have evolved over time and continue to evolve as new devices and capabilities are available. Initially, WSNs consisted of a flat topology made of homogeneous wireless sensor devices measuring a single variable. Most of these networks consisted of several (not many) wireless sensor devices spread throughout the area of interest and one sink node that received all the data and interfaced the WSNs with other networks so that WSN data could be made available at the remote site of interest for monitoring and analysis.

Then, newer applications required a considerable larger number of nodes, hundreds, or even thousands of them. These applications required new architectures for the efficient transmission of wireless sensor data. As such, layered or clustered topologies were designed that included multiple sinks. Further, these large networks required different points of connectivity with the outside world, so the interconnection of WSNs with the Internet, private networks, cellular networks, wireless ad hoc networks, etc., was a necessity. Of course, this triggered large efforts in the design and implementation of communication protocols to include these new capabilities. Figure 1.2 shows two flat small wireless sensor networks with a single sink node connected to a wireless ad hoc network, cellular network and the Internet, which at the same time can serve as interconnecting networks between them.

The incorporation of several sinks evolved network architectures to include heterogeneous devices. WSNs are now capable of measuring different variables, and able to take advantage of the capabilities of more powerful devices to perform complex functions of interest or even save additional energy. For example, more powerful devices have been given clusterhead roles to aggregate and transmit data on behalf of other less powerful nodes.

Finally, new advances in technology are producing and are expected to continue producing new devices that will soon reduce or eliminate most of the constraints that WSNs have today. Relaxing these constraints will expand the application domains of WSNs considerably. For example, research is being done today in what is called Wireless Multimedia Sensor Networks (WMSNs) assuming that several of these constraints no longer exist.

1.3 Application Domains and Examples

Wireless sensor networks have gained considerable popularity given their flexibility to solve problems in different application domains. Wireless sensor devices can be equipped with single or multiple sensors according to the application at hand. WSNs have been applied in a myriad of applications and have the potential to change our lives in many different ways. WSNs have been successfully applied in the following application domains:

- **Agriculture:** WSNs have been used to control irrigation systems according to the humidity of the terrain.
- **Military:** Intrusion detection systems based on WSNs have been used by the military.
- **Manufacturing:** WSNs have been used to monitor the presence of lethal gases in refineries.
- **Transportation:** Real-time traffic information is being collected by WSNs to later feed transportation models and alert drivers of congestion and traffic problems.
- **Environmental:** WSNs have been installed to monitor water deposits in mountains to detect mudslides. WSNs have been utilized in intelligent buildings to automatically control the temperature.
- **Engineering:** Civil engineers have used WSNs technology to monitor the condition of civil structures, such as bridges.

Additional advantages of WSNs are the possibility of monitoring these applications from remote places and having a system that can provide large amounts of data about those applications for longer periods of time. This large amount of data availability usually allows for new discoveries and further improvements.

1.4 Challenges and the Need for Energy Saving Mechanisms

Wireless sensor networks present a series of serious challenges that still need considerable research effort. While some of these challenges are a direct consequence of the constrained availability of resources in the wireless sensor nodes, and therefore very specific to WSNs, others are common challenges faced by most networking technologies. The following list briefly explains the most important challenges faced by WSNs today.

- **Network lifetime:** WSNs are battery powered, therefore the network lifetime depends on how wisely energy is used. In large scale wireless sensor networks or in dangerous applications it is important to minimize the number of times batteries need to be changed. It is desirable to have network lifetimes in the order of one or more years. In order to achieve such long network lifetimes it is imperative to operate the sensors in a very low duty cycle. For example, a typical energy consumption of a microcontroller is about 1 nJ per instruction. If the microcontroller is powered by a 1 J battery and the entire node is working continuously for one day, it needs to consume no more than 11 μW, which is not achievable by the microcontroller alone. Therefore, using the microcontroller and transceiver sleeping modes is crucial for the long operation of WSNs.
- **Scalability:** Some applications require hundreds or even thousands of wireless sensor devices. For example, imagine a WSNs to monitor the U.S.–Mexico border, or an application to monitor an oil pipeline. These large-scale WSNs present new challenges not seen in small-scale ones. Algorithms and protocols that work fine in small-scale networks don't work necessarily well in large-scale ones. One typical example is the routing function. Small-scale networks can easily run well-known proactive or reactive routing protocols using Dijkstra's shortest path algorithm. However, this approach will not be energy-efficient for large-scale wireless sensor networks. Location-based routing mechanisms using local information are better suited instead. Similar scalability problems arise in other areas.
- **Interconnectivity:** WSNs need to be interconnected so that data reaches the desired destination for storage, analysis, and possible action. WSNs are envisioned to be interconnected with many different networking technologies, as shown in Figure 1.2 before. However, this is easier said than done. New protocols and mechanisms need to be designed to achieve these interconnections and allow the transfer of data to and from WSNs. Normally, these interconnections are being handled by the use of gateway devices, such as the sinks, which require new capabilities for the appropriate discovery of networks and the translations of different communication protocols.
- **Reliability:** Wireless sensor devices are cheap devices with fairly high failure rates. Further, in many applications, these devices have to be thrown to the area of interest from a helicopter, or similar vehicle. As a result, several nodes break or partially break affecting their normal functionality. Node reliability is also effected by crucial levels of available energy.
- **Heterogeneity:** New WSNs are embedding wireless sensor devices with different capabilities and functionalities that require new algorithms and communication protocols. For example, cluster-based architectures may utilize more powerful devices to aggregate data and transmit information on behalf of resource constraint nodes. This heterogeneity includes the need of clustering and data aggregation algorithms that are not of trivial design.
- **Privacy and security:** Privacy and security are normal concerns in networking, and WSNs are not the exemption. However, security mechanisms are usually very resource demanding, which is not always in line with the resources avail-

able in wireless sensor devices. Therefore, new security algorithms with low computational complexity and low energy requirements are needed.

One constraint that somehow affects all these and other challenges is *energy*. Simply put, the lifetime of the network depends on the energy available in individual sensor nodes. Therefore, in WSNs, the design of energy-efficient algorithms and protocols is of utmost importance. It is not surprising that a large number of publications addressing energy concerns in Medium Access Control mechanisms, routing protocols, transport layer protocols, security, etc., exist.

Given this strong constraint, it is important to know which node components consume more energy to explore new design trade-offs. Normally, transceivers and microcontrollers are the most energy consuming components. However, one important aspect to consider is that both can be set to operate in less energy-operating modes whenever possible. However, given that they are working, which one consumes more energy? Typically, computing one instruction on a microcontroller consumes about 1 nJ. If we use the RFM TR1000 transceiver as an example, it needs 1 μJ to transmit a single bit and 0.5 μJ to receive one. Therefore, it is clear that communication is considerably more expensive than computing. In [98], the authors utilize example numbers from wireless sensor devices to reach the same conclusion pointing out that to reduce energy costs, it pays to process data locally to reduce the need of communication. This is the concept of *in-network processing*, or using computations to reduce communication costs. The authors also explain that if short-range communication is considered, as is usually the case in WSNs, the communication energy budget is dominated by the oscillators and mixers, which make transmitting as energy consuming as receiving.

Another interesting trade-off is found working with memory, FLASH memory in particular, since RAM memory consumption figures are usually included in the microcontroller figures. FLASH manufacturers data indicate that reading times and energy consumption figures for reading are similar regardless of the FLASH memory, however, writing times and writing energy consumption vary widely from memory to memory. Therefore, the difference in energy consumption between reading and writing operations can be considerable. In [80], the authors show that Crossbow's MICA motes consume 1.11 nAh reading and 83.33 nAh writing, a substantial difference that immediately suggests that it is better to avoid writing as much as possible.

Chapter 2
The Physical Layer

2.1 Introduction

This chapter covers some of the most important aspects of the Physical Layer of the protocol stack related to the study of WSNs. The chapter is not meant to cover available communication technologies, such as modulation techniques, noise and interference issues, and the like. Rather the chapter provides the reader with basic understanding of some fundamental concepts that are useful for later sections and chapters, such as wireless channel propagation models, energy consumption models, and sensing and error models in WSNs.

2.2 Wireless Propagation Models

This section provides an introduction to wireless propagation models. In wireless communications, signals travel from sender to receiver through the radio channel. These signals, which are sent at a particular power by the transmitter, suffer attenuation in the radio channel. The receiver, at the other end, is only able to receive the sender's transmission if the signal is received with a power level greater than the sensitivity of its transceiver. The attenuation, commonly known as the *path loss* of the channel, directly depends on the distance between sender and receiver, the frequency of operation and other factors. Path loss models exist to predict if there is a radio channel between two nodes. This section describes the three most commonly known path loss models available in the literature: the free space model, the two-ray ground model, and the log-distance path model [101].

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

2.2.1 The Free Space Propagation Model

The Friis' free-space propagation model applies when there is a direct and unobstructed path between sender and receiver, i.e., there is line-of-sight. The received power at a distance $d \geq d_0$ meters between sender and receiver is given by Friis' path loss equation 2.1.

$$P_{rx}(d) = \frac{P_{tx} \times G_{tx} \times G_{rx} \times \lambda^2}{(4\pi)^2 \times d^2 \times L} = C_f \times \frac{P_{tx}}{d^2} \tag{2.1}$$

where $P_{rx}(d)$ is the power received at the receiver over distance d, P_{tx} is the power at which the signal was transmitted, G_{tx} and G_{rx} are the gains of the antennae of the transmitter and receiver, respectively, $L \geq 1$ includes the losses in the transmitter and receiver circuits, λ is the wavelength in meters, and C_f is a constant that depends on the transceivers.

As it can be seen from Equation 2.1, the signal attenuates proportional to the square of the distance d that it has to travel. From the transmitter's point of view, Equation 2.1 says that a disk of radius $r = \sqrt{C_f \times P_{tx}}$ is created and centered at the node equivalent to its area of coverage.

2.2.2 The Two-Ray Ground Model

The two-ray ground model assumes a more realistic scenario in which the receiver receives signals that travel directly from sender to receiver and other signals that reach the receiver, as shown in Figure 2.1. This model is definitively more accurate than the Friis' free space model described before. The power received at the receiver over distance d is now given by Equation 2.2.

$$P_{rx}(d) = \frac{P_{tx} \times G_{tx} \times G_{rx} \times h_{tx}^2 \times h_{rx}^2}{d^4} = C_t \times \frac{P_{tx}}{d^4} \tag{2.2}$$

where h_{tx} is the height of the antenna of the transmitter, h_{rx} is the height of the antenna of the receiver, G_{tx} and G_{rx} are the gains of the antennae of the transmitter and receiver, respectively, P_{tx} is the power at which the signal is transmitted, and C_t is a constant that depends on the transceivers. As it can be seen from Equation 2.2, with the two-ray ground model, the signal now attenuates proportional to the fourth power of the distance d. Therefore, the transmitter has now a disk of radius $r = \sqrt[4]{C_t \times P_{tx}}$ equivalent to its area of coverage.

2.2.3 The Log-Distance Path Model

The log-distance path model was derived based on field measurements and curve fitting the collected data. This is a commonly used approach to derive path models in uncommon environments, such as caves or other places with abundant obstacles or reflecting materials. The log-distance model is given by Equation 2.3.

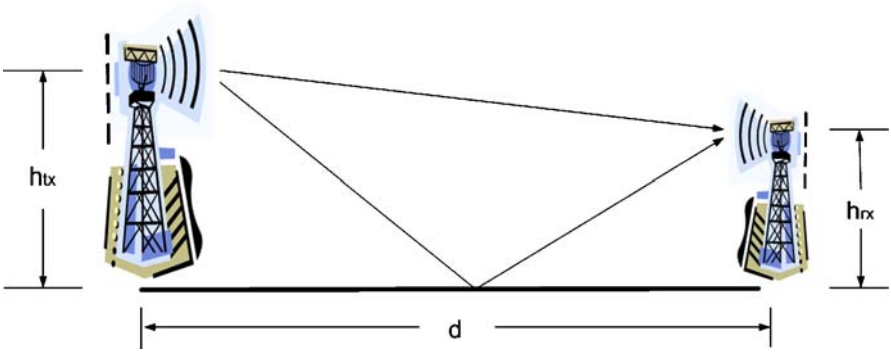


Fig. 2.1 The two-ray propagation model.

$$P_{rx}(d) \propto \frac{P_{tx}}{d^{\alpha}} \tag{2.3}$$

which says that the path loss is proportional to the transmission power P_{tx} and to the distance d between sender and receiver raised to the path loss exponent α that depends on the environment. In this case, the transmitter is able to have a disk of radius $r = \sqrt[\alpha]{P_{tx}}$ equivalent to its area of coverage.

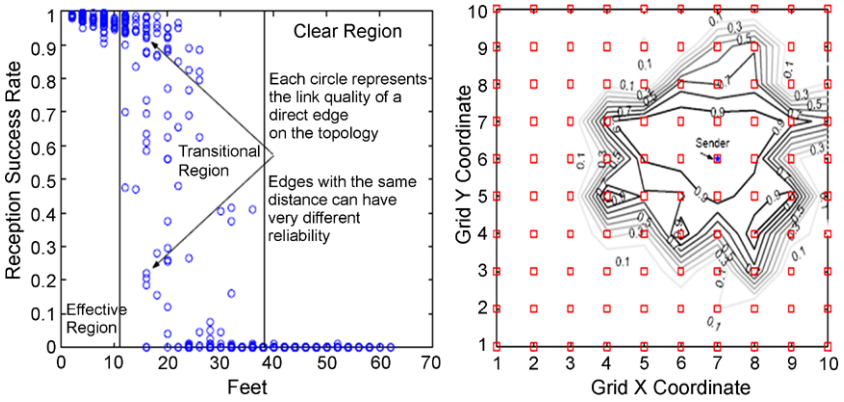
Equation 2.3 can be expressed in decibels as:

$$PL_{r,dB}(d) = PL_{dB}(d) + 10\alpha \, log\left(\frac{d}{d_0}\right) + X_{\sigma,dB} \tag{2.4}$$

where $PL_{r,dB}(d)$ is the received power in dB, $PL_{dB}(d)$ is the path loss in dB from sender to receiver over a distance d, d_0 is a reference distance, α is the path loss exponent, and $X_{\sigma,dB}$ is a zero-mean Gaussian random variable in dB with standard deviation σ.

Two interesting conclusions can be derived directly from Equation 2.4. First, the received power decreases with the frequency of operation. Second, the received power depends on the distance d according to a power law. A node at a distance αd to some receiver must spend α^2 times the energy of a node at distance d from the same receiver with the same received power P_{rx}.

Although analytical models and equations are good to predict the received power, empirical studies are important to validate the analytical results and better understand the specifics of a particular environment. One typical example is the assumption of a perfect disk area of coverage, as given by the equations above. An interesting empirical study of connectivity with a platform of MICA motes was carried out in [129] to show that connectivity is not binary but instead a probability of successful communication modified by fading, interferences, and other sources of loss. Figure 2.2(a) shows how nodes that are similar distances apart obtain very different packet reception rates, especially within the transitional region where individual pairs exhibit high variation. A link quality model with respect to distance was de-



(a) Packet reception rates over distance. (b) Area of coverage of a node.

Fig. 2.2 Empirical packet reception rates and area of coverage using an empirical path loss model [129] © 2003 ACM, Inc. Included here by permission.

rived from the observations in Figure 2.2(a) and later used to build a node's area of coverage, which definitively doesn't show a perfect circle (Figure 2.2(b)).

2.3 Energy Dissipation Model

Energy dissipation models are very important in WSNs, as they can be utilized to compare the performance of different communication protocols from the energy point of view. A very simple and commonly used energy dissipation model is the first order radio model introduced in [47]. The model, shown in Figure 2.3, considers that to transmit a k-bit packet from sender to receiver over a distance d, the system spends:

$$E_{Tx}(k,d) = E_{elec-Tx}(k) + E_{amp-Tx}(k,d) \tag{2.5}$$

$$E_{Tx}(k,d) = E_{elec} \times k + E_{amp} \times k \times d^2 \tag{2.6}$$

where $E_{Tx}(k,d)$ is the energy consumed by the transmitter to send a k-bit long packet over distance d, $E_{elec-Tx}(k)$ is the energy used by the electronics of the transmitter, and $E_{amp-Tx}(k,d)$ is the energy expended by the amplifier. Similarly, at the receiving node, to receive the message the transceiver spends:

$$E_{Rx}(k) = E_{elec-Rx}(k) \tag{2.7}$$

$$E_{Rx}(k) = E_{elec} \times k \tag{2.8}$$

where $E_{Rx}(k)$ is the energy consumed by the receiver in receiving a k-bit long packet, which is given by the energy used by the electronics of the receiver $E_{elec-Rx}(k)$.

In the same paper, the authors consider $E_{elec} = 50$ nJ/bit as the energy consumed by both the transmitter and receiver circuitry, $E_{amp} = 100$ pJ/bit/m^2, as the

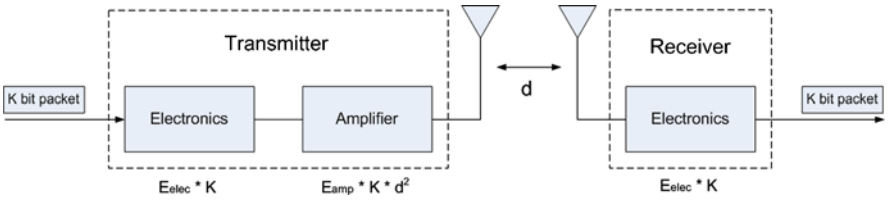


Fig. 2.3 The first order energy dissipation model introduced in [47].

energy consumed by the amplifier, and a path loss exponent $\alpha = 2$. For these numbers, which are typical values, it can be concluded that receiving packets is not a low cost operation; therefore, care must be taken in energy-efficient routing because paths with many short links may consume more energy than paths with fewer but longer links.

2.4 Error Models

Errors occur in different patterns and quantity according to the physical media and some other phenomena such as node speed, obstructions, multi-path, fading, etc. For example, in fiber optic-based channels, errors are considered to occur randomly and with very low rates, in the order of 10^{-9} or better. However, wireless channels are usually in the opposite side of the spectrum with bit error rates in the order of 10^{-6} or worse and errors occurring in "bursts". The manner errors occur and the quantity of errors have direct and important implications in the design and analysis of communication protocols and mechanisms, as it will be shown in later chapters. Two error models commonly used to model these two different scenarios and analyze the performance of communication protocols and mechanisms are the **independent error model** and the **two-state Markov error model**, which are described next.

2.4.1 The Independent Error Model

The first model, called the independent error model, assumes that errors occur at random and, therefore, they are independent. This model is "memoryless" as there is no temporal correlation between the symbols, i.e., the probability that one symbols is in error is not affected by what happened to any other symbol. This model is widely used because of its simplicity for mathematical analysis. Under the independent model, the probability that a frame of size L bits is received in error, P_e, is given by Equation 2.9 as:

$$P_e = 1 - (1 - p)^L \tag{2.9}$$

where p is the Bit Error Rate (BER) of the channel. This model is easy to understand and simple to use but it does not reflect the real behavior of several channels, and wireless channels in particular. In wireless channels, errors occur in "bursts" and, therefore, they are correlated.

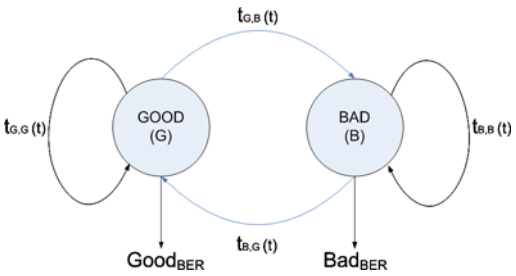


Fig. 2.4 Two-state Markov error model for wireless channels.

2.4.2 The Two-State Markov Error Model

A more appropriate model for wireless channels is the two-state Markov error model described in [122]. This model, illustrated in Figure 2.4, is implemented using a Discrete Time Markov Chain (DTMC) that models channel conditions at the bit level. During the "Good" state the channel is assumed to be in good condition with certain $Good_{BER}$. On the other hand, the "Bad" state indicates that the channel is experiencing a degradation in its quality providing a worse BER, Bad_{BER}, that produces a larger number of errors.

The final effect of the two-state Markov model is shown in Figure 2.5. Normally, the model stays more time in the Good state during which the channel exhibits a good quality. On the other hand, the chain stays in the Bad state for a rather short duration of time during which a higher BER is introduced. These long and short durations produce a "bursty" error model that reproduces common wireless effects, such as short fading, multi-path cancellations, etc., very accurately.

The two-state Markov model is characterized by four transition probabilities, the initial state probability distribution, and the error probability matrix. The transition probabilities indicate the probability of the chain of being in state $S = \{G(Good), B(Bad)\}$ at time $t + 1$ given that it was in state S at time t. Mathematically, these probabilities can be written as:

$$\begin{aligned} t_{G,G}(t) &= P\{S_{t+1} = G | S_t = G\} \\ t_{G,B}(t) &= P\{S_{t+1} = G | S_t = B\} \\ t_{B,G}(t) &= P\{S_{t+1} = B | S_t = G\} \\ t_{B,B}(t) &= P\{S_{t+1} = B | S_t = B\} \end{aligned} \tag{2.10}$$

which can be expressed in matrix form as:

$$T(t) = \begin{bmatrix} t_{G,G}(t) & t_{G,B}(t) \\ t_{B,G}(t) & t_{B,B}(t) \end{bmatrix} \tag{2.11}$$

which is called the *state transition matrix*.

It is well known that the state distribution Π_{t+k} at time $t+k$ can be easily found in an iterative manner using the state transition matrix as:

$$\Pi_{t+k} = \Pi_t \times T^k \tag{2.12}$$

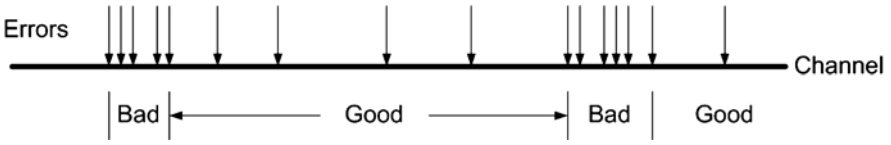


Fig. 2.5 Example of error pattern as produced by a two-state Markov error model.

where Π_t for $t = 0$ is the initial state probability distribution Π_0, which can be set to any value. The steady state distribution is then equal to:

$$\Pi_{ss} = \left[\pi_G \ \pi_B \right] \tag{2.13}$$

which is the same Π_{t+k} for a sufficiently large t and arbitrary value of k.

Finally, the *error probability matrix* E is defined as:

$$E = \begin{bmatrix} P\{C|G\} & P\{C|B\} \\ P\{M|G\} & P\{M|B\} \end{bmatrix} \tag{2.14}$$

where $P\{C|G\}$ is the probability that a Correct decision was made given that the chain was in the Good state, or the probability of having a good bit given that the channel was in the Good state; $P\{M|G\}$ is the probability of making a Mistake, or the probability of having a bad bit given that the channel was in the Good state; $P\{C|B\}$ is the probability of having a bad bit given that the channel was in the Bad state; and $P\{M|B\}$ is the probability of having a good bit given that the channel was in a Bad state.

By simple matrix multiplication, the probabilities of making a Correct decision or making a Mistake are calculated a follows:

$$\left[P_C \ P_M \right] = \Pi_{SS} \times E^T \tag{2.15}$$

where E^T is the transpose of matrix E.

Note that if the system is error free during the Good state, the well-known two-state Gilbert model is obtained, which was used by Gilbert to calculate the capacity of a channel with bursty errors [41].

An important question is, how can we derive the state transition matrix T and the error probability matrix E from either channel simulation results or real channel measurements, so that the two-state Markov chain accurately models the channel that originated those results or measurements? The answer to this questions is found in the well-known iterative procedure given by the Baum–Welch algorithm [8], which given a sequence of observations $O = \{O_1, O_2, \ldots, O_t, \ldots, O_T\}$, finds the maximum likelihood estimator $\Gamma = \{T, E\}$ that maximizes the probability that the sequence of observations were produced by the estimated parameters of the model T and E, $P\{O|\Gamma\}$. For the interested reader, all the necessary steps and software code needed to calculate and implement the Baum–Welch algorithm and, therefore, calculate the channel model Γ are very well detailed in [122].

Another way to calculate the probabilities of the two-state Markov model is given in [68, 101]. Under this method, the $P(Good)$, $P(Bad)$ and the transition probabilities of the DTMC $t_{G,B}$ and $t_{B,G}$ are calculated assuming a Raleigh fading channel. The method calculates the average number of positive crossings of the signal per second as:

$$N = \sqrt{2\pi} \times f_m \times \rho \times e^{-\rho^2} \tag{2.16}$$

where f_m is the maximum Doppler frequency and ρ is the Raleigh fading envelope normalized to the local rms level. The average time T during which the signal is below the threshold R is:

$$T = \frac{e^{\rho^2} - 1}{\sqrt{2\pi} \times f_m \times \rho} \tag{2.17}$$

With these equations, $P(Good)$ and $P(Bad)$ are calculated as follows:

$$P(Good) = \frac{1/N - T}{1/N} = e^{-\rho^2} \tag{2.18}$$

$$P(Bad) = \frac{T}{1/N} = 1 - e^{-\rho^2} \tag{2.19}$$

And the transition probabilities are then given by:

$$t_{G,B} = \frac{N}{R_t \times P(Good)} \tag{2.20}$$

$$t_{B,G} = \frac{N}{R_t \times P(Bad)} \tag{2.21}$$

where R_t is the transmission rate in symbols per second of the communication system under consideration.

Using these results, the final average BER can be calculated as the summation of the probability of being in either state times the BER of each state as:

$$BER = Good_{BER} \times P(Good) + Bad_{BER} \times P(Bad) \tag{2.22}$$

2.5 Sensing Models

Similar to the communication coverage of a node, which is determined by the transmission power of the transceiver and the propagation models discussed before, sensors have a their own *sensing coverage*. An important aspect to emphasize here is that the sensing coverage does not necessarily match the communication coverage, therefore ensuring connectivity and communication coverage does not mean that the network also ensures sensing coverage. There are two widely known sensing models utilized in the literature to model the sensing coverage: the **binary sensing model** and the **probabilistic sensing model**.

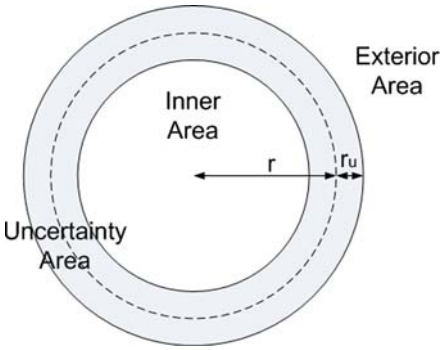


Fig. 2.6 The probabilistic sensing model.

2.5.1 The Binary Sensing Model

The binary sensing model assumes that the sensing coverage of a sensor s, $C(s)$, is given by a disk with a fixed radius r. Any event that occurs at a point p in the 2-dimensional plane, (x_p, y_p), will be detected by sensor s, if the Euclidean distance between s and p, $d(s, p)$, is within the sensing range of s, as follows [147]:

$$C(s) = \begin{cases} 1, & \text{if } d(s, p) \leq r \\ 0, & \text{otherwise} \end{cases} \tag{2.23}$$

According to Equation 2.23, the sensor will detect event e with probability 1, if $d(s, p)$ is less than or equal to r, and will not detect it otherwise. This deterministic model is not very realistic given the imprecise sensor detections. In reality, the sensor coverage is not a perfect circle and the coverage is modeled using a probabilistic model.

2.5.2 The Probabilistic Sensing Model

Under the probabilistic sensing model, sensors have three very well defined areas, as shown in Figure 2.6. The inner area is given by the distance $r - r_u$, which ensures that the sensor will detect event e with probability 1, where r_u is known as the uncertainty distance. The exterior area is just the opposite. This is the part where the probability of sensing event e is zero. In other words, $d(s, p)$ is greater than $r + r_u$. Finally, there is the uncertainty area (gray area in the figure) in which the sensor will detect event e with certain probability that decays exponentially with the distance. Mathematically, sensor s sensing coverage can be expressed as [147]:

$$C(s) = \begin{cases} 1, & \text{if } r - r_u \geq d(s, p) \\ e^{-\lambda \alpha^{\beta}}, & \text{if } r - r_u < d(s, p) \leq r + r_u \\ 0, & \text{if } r + r_u < d(s, p) \end{cases} \tag{2.24}$$

where $\alpha = d(s, p) - (r - r_u)$ and β and λ are parameters that yield different detection probabilities that can be used to model different types of physical sensors, in particular, range sensing devices, such as infrared and ultrasound sensors.

Chapter 3
The Data Link Layer

3.1 Introduction

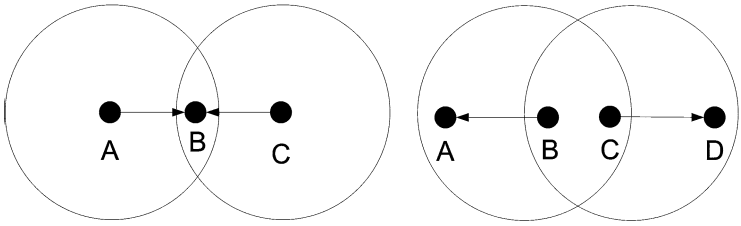
This chapter describes the most important Data Link Layer (DLL) algorithms and protocols for WSNs with emphasis on those with energy-saving features. The Data Link Layer, which is the second layer in the communication protocol stack, has two sub-layers: the Medium Access Control (MAC) sub-layer, which sits on top of the Physical Layer, and the Logical Link Control (LLC) sub-layer, which interfaces the DLL with the upper layers. This chapter describes the main performance goals and mechanisms in each sub-layer, as they pertain to wireless sensor networks.

3.2 The Medium Access Control Sub-layer

Medium Access Control protocols have the responsibility of providing fair access to all nodes sharing a common communication channel while achieving good individual throughput and overall channel utilization. In traditional MAC protocols for wired networks, these goals were mainly achieved by mechanisms meant to detect and deal with collisions in a fast and effective manner. The best example is the well-known and widely-used Carrier Sense Multiple Access with Collision Detection (CSMA/CD) mechanism included in the IEEE 802.3 standard [55] that allows stations to hear for collisions while transmitting their frames. Although the MAC sub-layer has the same performance goals in wireless communications, MAC protocols for wired networks had to be modified to deal with new specific issues related to the wireless media. For example, the CSMA/CD protocol does not perform well in wireless networks because of its inefficiency in dealing with two unique problems in wireless networks: the *hidden terminal problem* and the *exposed terminal problem*.

The hidden terminal problem is depicted in Figure 3.1(a), which shows the failure of the carrier sense mechanism of the CSMA protocol. If node A is transmitting

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,



(a) The hidden terminal problem. (b) The exposed terminal problem.

Fig. 3.1 The hidden and exposed terminal problems in wireless communications.

to node B and node C wants to transmit to B, it will be able to do so, as it is unable to detect A's transmission, i.e., A and C are hidden to each other. As a result, a collision at B will occur. This phenomena occurs because of the attenuation that wireless signals face over distance. The exposed terminal problem, depicted in Figure 3.1(b), shows how the CSMA wastes channel bandwidth by deferring transmissions unnecessarily. If node B is transmitting to node A, and node C wants to transmit to D, node C will be unable to acquire the channel because the carrier sense mechanism will find the channel busy. These two problems occur mainly because the sensing responsibility is given to the transmitting node. In wireless networks, sensing at the receiver along with some form of communication between sender and receiver is needed to solve these problems. Protocols that work in this manner will be introduced later.

Although wireless sensor networks are also wireless networks facing the same issues, the MAC sub-layer in WSNs has different goals and performance concerns than other wireless networks:

- Wireless sensor network devices usually send very small frames and use the channel occasionally, either periodically or whenever an important event occurs. As a result, fairness is not as important as in other networks where there is the chance that one node would monopolize the use of the channel. In WSNs nodes rarely compete for the channel.
- WSNs are normally data-oriented networks without hard delay requirements. Fast channel access, although desirable, is not an important issue either.
- Achieving good channel utilization is not as important, as nodes are idle most of the time.
- In WSNs energy is the most important issue, therefore, additional mechanisms to save energy are of utmost relevance. In most designs, fairness, fast access to the channel (delay), throughput and other relevant metrics in other networks are traded for energy conservation. In particular, MAC sub-layer protocols for WSNs must address the following energy-related issues:
  - **Collisions:** Collisions should be avoided because of the extra energy wasted in frame retransmissions. Recall from Chapter 1 that communication is the most energy spending function.

- **Overhead:** Control messages and long headers in frames need to be avoided as much as possible, as they imply extra expensive communication costs.
- **Overhearing:** Overhearing is the energy consumed by the nodes by being constantly listening and decoding frames that are not meant for them. This is a consequence of using a shared media in which nodes do not know a priori whether the transmissions are for them or not.
- **Idle listening:** Idle listening refers to the energy expended by the nodes by having their circuits on and ready to receive while there is no activity in the network. This is particularly important in WSNs, as nodes use the channel sporadically. Strategies to turn nodes on and off are very important in WSNs.
- **Complexity:** Complexity refers to the energy expended as a result of having to run computationally expensive algorithms and protocols. One of the most important design goals in WSNs is therefore simplicity.

3.2.1 Common MAC Protocols

The following is a brief review of the taxonomy and most important MAC protocols utilized today. Figure 3.2 is just one attempt to categorize the large number of MAC protocols available. Furthermore, the list of protocols included is by no means an exhaustive list. In the figure, MAC protocols are classified according to how they assign the shared channel to the users: *fixed assignment*, *demand assignment*, and *random assignment*.

In **fixed assignment MAC protocols**, as its name implies, the MAC protocol assigns a fixed portion of the shared channel to each user. An important performance aspect of these methods is that the assignments are established without any consideration as to whether or not the user is actually using the channel. Also, fixed assignment MAC protocols usually need a central brain to determine and control the assignments.

From a wireless sensor network point of view, fixed assignment methods have advantages and disadvantages. The centralized architecture is difficult to implement in those scenarios with flat topologies made of a large number of sensors. Cluster-based topologies, on the other hand, may benefit from a fixed assignment MAC protocol, especially if clusterheads have enough computational, communication, and energy capabilities. Further, a fixed assignment method will be easy to integrate with sleep and wake-up procedures of the nodes, as the central station and the nodes know when each node is scheduled to transmit or receive information. Another advantage of fixed-assignment methods is that collisions do not occur, avoiding the additional energy costs involved in retransmissions. Time-Division Multiple Access (TDMA), Frequency-Division Multiple Access (FDMA) and Code-Division Multiple Access (CDMA) are very widely-used fixed assignment methods in cellular networks. These protocols are very well-known; therefore, they will not be described any further [101].

Random assignment MAC protocols are the most flexible to implement because they work on a completely distributed manner; they do not have the need of

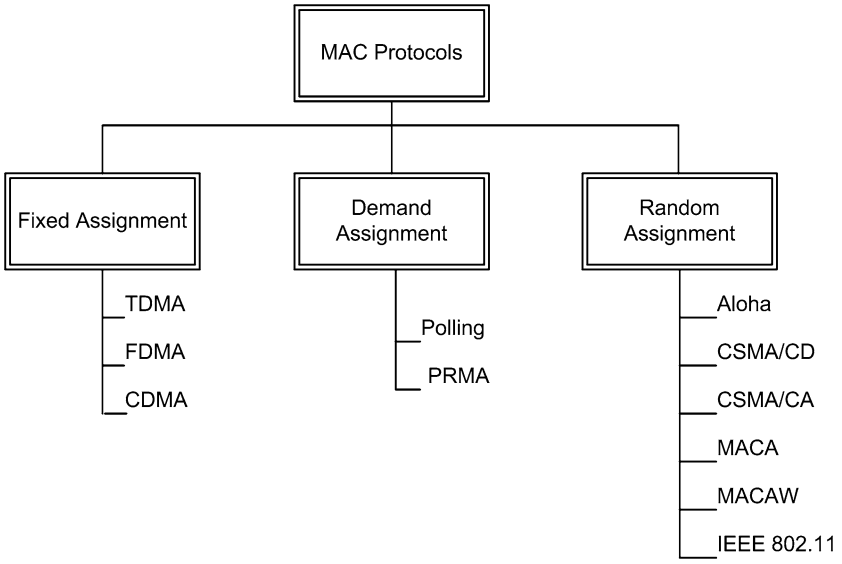


Fig. 3.2 Taxonomy of Medium Access Control protocols.

any central brain or controller. As the name implies, randomness is a critical component in the assignment of the channel. Nodes try to obtain the channel when they need it – they compete for the channel. As a result, collisions are possible, therefore, random assignment MAC protocols must include mechanisms to detect and recover from collisions.

Random assignment protocols are very well suited for wireless sensor networks not only because of its distributed nature but also because of their efficiency transmitting data from applications like the ones in WSNs, i.e., small and sporadic frame transmissions. On the other hand, these protocols normally do not solve the overhearing and idle listening problems explained before.

Several well-known protocols exist in this category. The Aloha protocol [1] developed at the University of Hawaii in 1970 for data transmissions over satellite channels was the pioneer protocol using random assignments. Under the Aloha protocol, nodes are allowed to send frames at any time without inspecting the channel at all. Although extremely simple, the Aloha protocol was shown to be very inefficient. Under some general assumptions, it has been shown that the Aloha protocol achieves a maximum channel utilization of 18%. The widely-used Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol of the IEEE 802.3 standard [55] improves the channel utilization over the Aloha protocol by incorporating the carrier sense capability. As explained before, this carrier sense capability was not very helpful avoiding collisions in wireless networks; modifications were needed to improve the performance of the protocols and also to avoid the hidden and exposed terminal problems. As a result, the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol was introduced.

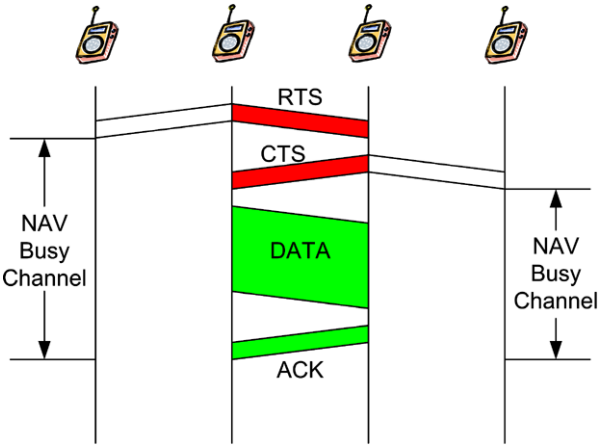


Fig. 3.3 The MACA protocol with ACK and the Network Allocation Vector (NAV).

In the CSMA/CA MAC protocol, nodes first sense the channel and, if idle, they transmit a small control frame, the Request To Send (RTS) frame, to request the channel. If the destination is idle, it will respond with another small control frame, the Clear To Send (CTS) frame, to tell the sender that data transmission can begin. The collision avoidance part comes from the fact that all nodes surrounding the sender and the receiver, upon receiving the RTS and CTS frames, are prohibited from transmitting for a long enough period of time so that collisions are avoided. Figure 3.3 shows the exchange of control and data frames and the Network Allocation Vector (NAV), an internal timer set by the respective nodes during which they are not allowed to use the channel. This procedure solves the hidden terminal problem in most of the cases. Collisions are not completely avoided by this method though. RTS frames from two nodes trying to reserve the channel at the same time will unavoidable collide. There is also the possibility of CTS frames colliding with data frames [112]. However, in all these cases, the advantage is that control frames are very small and the channel is not wasted as much as if long data frames had collided. The protocol also includes a random mechanism for nodes to re-schedule their transmissions in case of collisions, or in those cases where the node senses the channel busy.

Several other protocols in this category exist, mainly based on enhancements or modifications to the CSMA/CA protocol. The MACA protocol proposed in [61] is an important one. MACA avoids collisions and solves the difficult problem of determining the status of the channel including information related to the length of the transmissions in RTS and CTS frames (time that the acquiring node will hold the channel). Nodes are constantly looking at RTS and CTS frames to know how long they have to wait without transmitting. MACA was also modified several times to improve its performance. The MACAW protocol described in [9] includes the ACK frame after the RTS–CTS-data transmission sequence to address throughput degradations derived from the recovery of those frames at the transport layer. The

Floor-Acquisition Multiple Access (FAMA) protocol [37] is an improvement over the MACAW protocol. The MAC layer included in the IEEE 802.11 standard [52] for wireless local ares networks is also derived from MACA and its subsequent improvements.

Demand assignment MAC protocols try to overcome the limitations of fixed assignment methods assigning unused channels to needed users on demand. With few exceptions, these protocols require a central station that receives demands and assigns resources according to availability and other criteria. Polling mechanisms belong to this category. In polling, the central station asks the nodes if they need the channel following a particular and predefined order. Nodes that do not have anything to transmit decline the poll while nodes that need the channel inform the central station, which later assigns the channel. Polling mechanisms are very old; they were used in IBM SNA networks in the 1980s. One of the most recent versions of a polling mechanism is the one found in the IEEE 802.15.1 Wireless Personal Area Network standard [54] based on the Bluetooth ™Foundation Specifications.

Other demand assignment protocols utilize fixed and random assignment protocols to reserve the channel prior to data transmission. Normally, time is divided into frames that have a portion reserved for the reservations and another portion for data transmission. Nodes utilize either fixed or random assignment protocols during the reservation period to send their data transmission requirements to the central station. Upon receiving all demands, the central station then assign slots of the data transmission period to the requesting nodes. The Packet Reservation Multiple Access (PRMA) protocol [43] is one example of a demand assignment protocol. PRMA uses the Aloha protocol to make reservations and a TDMA-like protocol for data transmissions.

3.2.2 MAC Protocols for WSNs

The MAC protocols described so far are not well-suited for WSNs as they were designed to work on computers and networks without the limitations and constraints of WSNs. For example, none of these protocols were designed with energy-efficiency in mind and were mostly concerned about throughput and fairness issues. As a result, a large number of MAC protocols for WSNs appeared in the literature over the last few years. This section describes a number of energy-efficient MAC protocols for WSNs and states their contributions toward addressing their main issues such as overhearing, idle listening, and collision avoidance. The protocols fall within the same categories explained before, as shown in Figure 3.2.

Random assignment or contention-based protocols are included first. They are more amenable to WSNs because of their distributed nature and scalability. Examples of protocols that belong to this category are PAMAS, S-MAC, T-MAC, and B-MAC protocols briefly described next.

The **Power Aware Multi-Access with Signaling (PAMAS) protocol** [112] is based on the MACA protocol but includes a separate signaling channel to avoid the collisions and overhearing problems. In the PAMAS protocol, all nodes utilize the signaling channel to exchange RTS–CTS frames and therefore gain access to the

media. As a result, all nodes know who has gained the media and for how long, information that nodes use to turn themselves off. The other channel is used exclusively to transmit data frames, which is collision-free. The main disadvantage of PAMAS is that it needs an additional radio for the signaling channel, which adds to the cost of sensor network devices.

The **Sensor MAC (S-MAC) protocol** was introduced in [140, 141] to solve the energy consumption related problems of idle listening, collisions, and overhearing in WSNs using only one transceiver. S-MAC considers that nodes do not need to be awake all the time given the low sensing event and transmission rates. S-MAC reduces the idle listening problem by turning the radio off and on periodically. Nodes are synchronized to go to sleep and wake up at the same time. In order to address the issue of synchronization over multi-hop networks, nodes broadcast their schedules to all its neighbors. This is performed sending a small SYNC frame with the node schedule periodically.

S-MAC divides time in two parts: the active (listening) part and the inactive (sleeping) part. The active part is divided at the same time in two time slots. During the first time slot, nodes are expected to send their SYNC frames to synchronize their schedules. The second time slot is for data transmission in which the S-MAC protocol transmits all frames that were queued up during the inactive part. In order to send SYNC frames over the first time slot or RTS–CTS–DATA–ACK frames over the second time slot, nodes obtain access to the media utilizing the same contention mechanism included in IEEE 802.11, which avoids the hidden terminal problem and does a very good job avoiding collisions too. However, nodes using the IEEE 802.11 protocol waste a considerable amount of energy listening and decoding frames not intended for them. In order to address this problem, S-MAC allows nodes to go to sleep after they hear RTS or CTS frames. During the sleeping time, a node turns off its radio to preserve energy.

The **Timeout-MAC (T-MAC) protocol** [26] introduces the idea of having an adaptive active/inactive (listening/sleeping) duty cycle to minimize the idle listening problem and improve the energy savings over the classic CSMA and S-MAC fixed duty cycle-based protocols. Therefore, the active part of the T-MAC protocol is variable (see Adaptive Time in Figure 3.4) and its length is dynamically determined by means of an active timer that is calculated in a very simple manner. The protocol times out when it does not hear anything else. The T-MAC protocol, however, suffers from the known *early sleep* problem, which can reduce throughput. Figure 3.4 shows the difference between the S-MAC and the T-MAC protocols [26].

The **Berkeley Media Access Control (B-MAC) protocol** [97] is a CSMA-based MAC protocol for WSNs. B-MAC introduces several interesting mechanisms and features. One mechanism is the Clear Channel Assessment (CCA) for effective collision avoidance, which takes samples of the media to estimate the noise floor. When a node wants to transmit, it takes a sample of the channel and compares it with the noise floor. If the sample energy level is substantially below the noise floor, the channel is said to be clear. B-MAC also utilizes a preamble sampling-like technique called Low Power Listening (LPL) to minimize the idle listening problem. Finally, B-MAC includes the use of ACK frames for reliability purposes and throughput im-

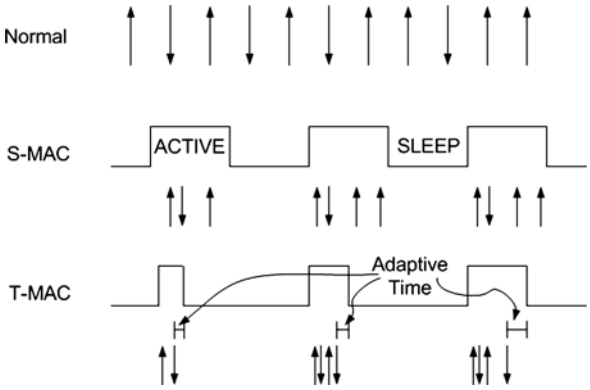


Fig. 3.4 The S-MAC and T-MAC protocols.

provement. One of the most interesting features of B-MAC not available in any other protocol thus far is the capability of tuning its operation and mechanisms. B-MAC provides interfaces that can include/exclude B-MAC mechanisms, such as the CCA, acknowledgments, and LPL to trade-off power consumption, latency, throughput, fairness or reliability.

Fixed assignment protocols are also utilized in WSNs mainly because their ability to avoid collisions and setup sleeping schedules. Here, the WiseMAC, LEACH and SMACS protocols are briefly described as examples of protocols that belong to this category.

The ***Wi*reless *Se*nsor MAC (WiseMAC) protocol** [34] is included in this category because of its centralized nature, as it takes advantage of the energy-unconstrained access point to achieve energy efficient communications in the downlink. In order to minimize idle listening and therefore save energy, WiseMAC utilizes a preamble sampling technique similar to the one described in [33]. In the preamble sampling technique, nodes sample the medium periodically to check for activity. If the medium is busy, nodes continue to listen until the data frame is received or until the channel becomes idle again. The access point appends data frames with a preamble of size equal to the sampling period to guarantee that the nodes will be awake when the data portion arrives. This technique is shown in Figure 3.5. The WiseMAC protocol improves the performance of the preamble sampling technique by minimizing the length of the preamble, which is a source of energy and channel utilization wastage. In WiseMAC the nodes let the access point know their sampling schedules and the access point uses this information to send the data frames at the right time with a minimum preamble.

The **Low Energy Adaptive Clustering Hierarchy (LEACH) protocol** [47] includes several ideas to reduce the energy consumption in WSNs. First, LEACH includes the idea of clustering but without including any powerful node as the clusterhead. A randomized rotation mechanism is included that rotates the clusterhead position and therefore distributes the energy among the nodes in an evenly manner. Second, in order to save additional energy in the nodes, LEACH utilizes a TDMA

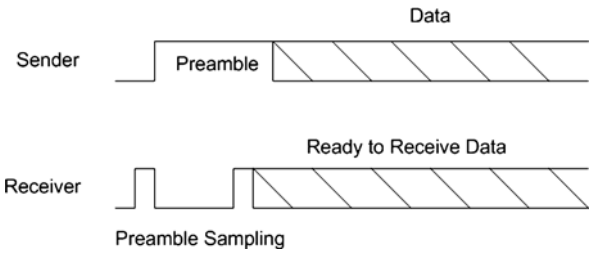


Fig. 3.5 The preamble sampling technique used in WiseMAC.

schedule-based MAC mechanism for intra-cluster communications. As explained before, a TDMA MAC protocol avoids collisions, the hidden and exposed terminal problems, and overhearing and idle listening problems by allowing nodes to turn themselves on and off at appropriate times (given by the schedule). In LEACH there is no inter-cluster communications, instead clusterheads are meant to transmit directly to the sink node. In order to avoid collisions, LEACH utilizes CDMA for clusterhead–sink communication, so clusterheads can transmit simultaneously to the sink without colliding with each other. Finally, LEACH performs local data fusion or data aggregation at the clusterheads. All these strategies combined allow the LEACH protocol to save considerable energy by eliminating the transmission of repeated information or by aggregating data from several sensors in one frame.

Although the LEACH protocol was shown to extend the network lifetime and distribute the energy consumption evenly among the sensors, it has some drawbacks. First, the clusterheads have to perform very computationally difficult and energy-consuming tasks, such as preparing and managing the TDMA schedule, fusing and aggregating the data, and transmitting directly to the sink node. Second, LEACH needs synchronization so that the TDMA scheme can actually work. Lastly, LEACH lacks multi-hop routing capabilities, which limit its applicability to small spaces. This last issue has been addressed in similar protocols, such as the HEED protocol presented in [142].

The **Self-Organizing Medium Access Control for Sensor Networks (SMACS) protocol** was introduced in [114] as the protocol in charge of network startup and link layer organization in a series of protocols proposed to perform organization, routing, and mobility (ORM) functions in wireless sensor networks. SMACS is a distributed infrastructure-building protocol based on a neighborhood discovery procedure and the establishment and exchange of transmission schedules. When two nodes discover each other, they agree to establish communication over a pair of fixed time slots periodically using a frequency chosen from a pool of possible choices. Similarly, other nodes do the same but because there are many frequencies from which to choose, and frequencies are chosen uniformly at random, the probability of choosing the same frequency is very small. In this manner, SMACS, which is a variation of a hybrid TDMA/CDMA-based access protocol, achieves collision-free communications. This is the main reason why, although a distributed protocol, SMACS is included in the fixed-assignment category.

Demand assignment protocols are not as popular in WSNs as their counterparts, mainly because of their hybrid nature, which usually adds to the complexity of the node. Here, only the TRAMA protocol is described given its distributed nature, which eliminates the need of a central arbiter and the single point of failure problem.

The **TRaffic-Adaptive Medium Access (TRAMA) protocol** [100] is a distributed TDMA mechanism that allows for flexible and dynamic scheduling of time slots. TRAMA consists of three components that assign time slots only to stations that have traffic to send while being implemented in a distributed fashion. In other words, TRAMA provides the energy-saving advantages of schedule-based mechanisms without the disadvantages of having a node as the main controller. Further, it performs better than contention-based mechanisms because the slot assignment avoids collisions. TRAMA nodes use the Neighbor Protocol (NP) and the Schedule Exchange Protocol (SEP) to send their transmission schedules along with information related to the current time slot, one and two-hops away node identifiers, and traffic interests. Then, the Adaptive Election Algorithm (AEA) uses this information to determine the transmitters and receivers for the current time slot and derive the node's sleep schedule. TRAMA utilizes the idea of node identifiers included in the **Node Activation Multiple Access (NAMA) protocol** [5] but extends NAMA's capabilities to improve the energy efficiency needed in WSNs.

3.3 The Logical Link Control Sub-layer

The LLC sub-layer sits on top of the MAC sub-layer and it is in charge of achieving good link utilization and providing a reliable service to the layers above despite the time-varying quality conditions of wireless links. In order to achieve these goals, the LLC has to perform the following functions [116, 49]:

- **Framing:** Data units must be encapsulated into LLC sub-layer frames to add necessary information to perform the sub-layer's functions. In WSNs, frames should be kept to the minimum size to save energy.
- **Error control:** Error detection and error correction mechanisms must be embedded to detect and correct bit errors introduced in the media. The energy efficiency of error control mechanisms highly depends on the characteristics of the media and how errors occur.
- **Flow control:** The LLC sub-layer includes mechanisms to control the flow of information and avoid overflowing the receiver. Given the low data rates of WSNs applications flow control is an issue of secondary importance. Normally, flow control mechanisms are embedded into error control mechanisms.
- **Link management:** Neighbor discovery, link setup, maintenance and tear down, and link quality estimation are among the most important link management functions performed by the LLC. Neighbor discovery and link quality estimation are functions utilized by many routing and topology control protocols to save energy.

Of all these functions, framing, flow control and link management are standard mechanisms very well covered in the networking literature, therefore, the reader

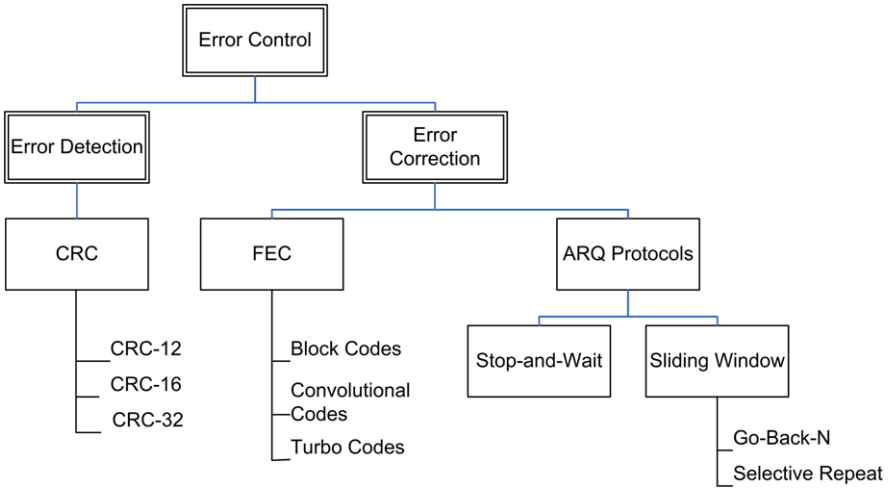


Fig. 3.6 Error control mechanisms.

is referred to the standard literature. The rest of this section concentrates on error control mechanisms since they have important reliability and energy consumption implications for WSNs. The most important error control mechanisms are described in detail next.

3.3.1 Error Control

Error control refers to those mechanisms that detect and correct errors that occur during the data transmission. Figure 3.6 shows the taxonomy that will be used to guide the description of these mechanisms.

Error detection mechanisms are normally based on Cyclic Redundancy Check (CRC) codes, which are some of the most common and powerful error-detecting codes. In addition, CRC codes are very simple and efficiently implemented in hardware. Well-known CRCs are CRC-12, CRC-16 and CRC-32 – this last one utilized in IEEE 802 Local Area Network standards. Error correction is usually accomplished by Forward Error Correction (FEC) mechanisms, Automatic Repeat Request (ARQ) protocols, and combinations of the two. These mechanisms are described in more detail next.

Forward Error Correction mechanisms are based on the addition of redundant information to the LLC frames. This additional information is utilized by the receiver to correct some bit errors. Widely-known FEC mechanisms are block codes, convolutional codes and more recently, turbo codes. In block codes and convolutional codes, k bits of user data are mapped to n channel symbols where $n \geq k$. Block codes are coded independent of each other but convolutional codes are not. Popular and widely used block codes are Reed–Solomon (RS) codes and Bose–Chaudhuri–Hocquenghem (BCH) codes. In [104] block codes and convolutional

codes are compared having energy efficiency as the main optimization metric. Under the assumption of independent bit errors, it is shown that the BCH code outperformed the best convolutional code by almost 15%. Turbo codes have received a lot of attention but not in WSNs due to their complex implementation.

Automatic Repeat Request (ARQ) protocols are classified as Stop-and-Wait and Sliding-Window protocols. The **Stop-and-Wait protocol** is the simplest of all. This protocol is widely used in LLC protocols designed for half duplex operation, such as several wireless networks. In the Stop-and-Wait protocol, the sender sends a data frame and waits for an acknowledgment before sending the following frame. If for any reason the acknowledgment is never received, the sender times out and retransmits the frame. Since the LLC works on a point-to-point manner, the value of this timeout can be easily and accurately estimated. Stop-and-Wait, although very simple to implement, has important drawbacks in certain scenarios. For example, it is easy to realize that a connection sending data using this protocol over a satellite link will experience very low throughput, as the channel will be idle most of the time.

Sliding-Window protocols are meant to improve the efficiency of Stop-and-Wait by permitting the sender to transmit a window with W frames before receiving any feedback from the receiver. As such, if an appropriate value of W is chosen, these protocols would be able to "fill the pipe" entirely with data, achieving 100% utilization. In the case of LLC protocols, choosing W is straight forward because the length of the channel is fixed and the channel's capacity is known. The value of W that would achieve 100% channel utilization is given by the *bandwidth–delay product* of the link, where delay means the round-trip time delay. In the case of a transport layer protocol, variable delays experienced at the queues in the routers along the path make this calculation more difficult.

Sliding-window protocols are not as simple to implement as the Stop-and-Wait protocol because they need to keep track of which frames are being transmitted and acknowledged in order to delete them from memory. There are two known sliding-window protocols. The **Go-Back-N protocol** sends a window W worth of frames and waits for acknowledgments. If the receiver detects an error or a missing frame, it will ask for a retransmission and will drop all subsequent ones. The sender will then retransmit all frames from that particular frame on, regardless of whether the subsequent frames were received correctly or not by the receiver. The complexity of the Go-Back-N protocol resides at the sender as the receiver can just drop all subsequent frames after it notices a hole in the sequence.

In the **Selective Repeat protocol**, when the receiver asks for the retransmission of lost frames the sender only retransmits the requested frames. This adds extra complexity in the protocol, in particular at the receiver, which now has to keep in memory all received frames from the same window until the missing frames are correctly received.

3.3.2 Performance Analysis of LLC Protocols

The efficiency of the ARQ protocols has been widely studied in the literature using the independent error model described in Section 2.4. Here, we follow the analysis included in [116].

If we assume that t_{frame} is the frame transmission time given by the frame size divided by the link speed, t_p is the propagation delay of the link, given by the length of the link divided by the speed of the transmission media, and neglect the transmission time of the acknowledgments and the processing time at the receiver, it can be easily seen that the efficiency η of the Stop-and-Wait protocol is given by Equation 3.1.

$$\eta = \frac{t_{frame}}{2t_p + t_{frame}} = \frac{1}{1 + 2a} \tag{3.1}$$

where a is equal to the propagation delay over the transmission time of the frame ($a = t_p / t_{frame}$).

Equation 3.1 provides the efficiency of the Stop-and-Wait protocol in an error-free environment. A more realistic calculation of the efficiency of the protocol must therefore include channel errors. If p is the BER of the channel and we consider the independent model, then Equation 2.9 provides the frame error probability. Therefore, the probability that i transmissions will be required before the frame is received correctly corresponds to a random variable geometrically distributed given by:

$$P[i \text{ transmissions}] = P_e^{i-1}(1 - P_e) \tag{3.2}$$

with mean value N equal to:

$$N = \sum_{i=1}^{\infty} i P_e^{i-1}(1 - P_e) = \frac{1}{1 - P_e} \tag{3.3}$$

So, the resulting efficiency of the Stop-and-Wait protocol considering independent errors is given by the error free efficiency reduced by the mean number of times a frame has to be retransmitted until received correctly:

$$\eta = \frac{\frac{1}{1+2a}}{N} = \frac{1 - P_e}{1 + 2a} \tag{3.4}$$

A similar analysis can be carried out to analyze the performance of the sliding-window protocols. It is easily seen that the efficiency of the Go-Back-N and the Selective Repeat protocols in an error-free environment is given by:

$$\eta = \begin{cases} 1, & \text{if } W \geq 2a + 1 \\ \frac{W}{2a+1}, & \text{if } W < 2a + 1 \end{cases} \tag{3.5}$$

As in the case of the Stop-and-Wait protocol, errors must be included in the final equations. In the case of the Selective Repeat protocol, since it only retransmits the frames in error, the rationale used in the case of the Stop-and-Wait protocol can

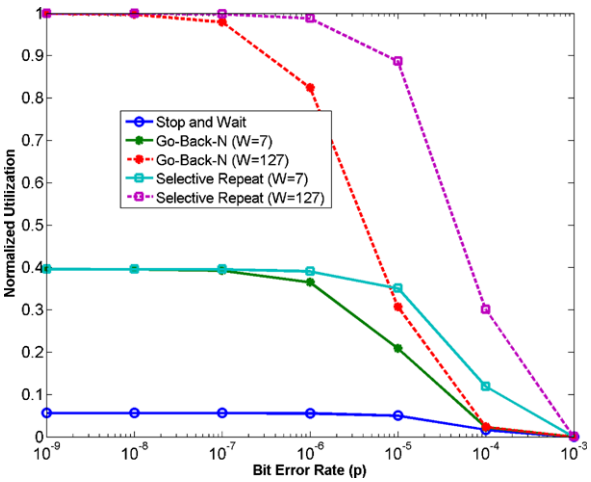


Fig. 3.7 Throughput of ARQ protocols.

be used here as well, i.e., it is enough to divide the error-free efficiency by the average number of transmissions that are necessary to receive the frames correctly (Equation 3.3). As a result, the efficiency of the Selective Repeat protocol is given by:

$$\eta = \begin{cases} 1 - P_e, & \text{if } W \geq 2a + 1 \\ \frac{W(1-P_e)}{2a+1}, & \text{if } W < 2a + 1 \end{cases} \tag{3.6}$$

In the case of the Go-Back-N sliding window protocol, an additional calculation must be made to include not only the expected number of transmissions but also the average number of frames that are retransmitted in each retransmission. Therefore, Equation 3.3 now has to be rewritten as:

$$N = \sum_{i=1}^{\infty} f(i) P_e^{i-1}(1 - P_e) = \frac{1 - P_e + K P_e}{1 - P_e} \tag{3.7}$$

where $f(i)$ is the total number of frames transmitted if the original frame must be sent i times, and K is approximately equal to $(2a+1)$ for $W \geq (2a+1)$, and $K = W$ for $W < (2a+1)$. Therefore, the final efficiency of the Go-Back-N protocol is given by:

$$\eta = \begin{cases} \frac{1-P_e}{1+2aP_e}, & \text{if } W \geq 2a + 1 \\ \frac{W(1-P_e)}{(2a+1)(1-P_e+WP_e)}, & \text{if } W < 2a + 1 \end{cases} \tag{3.8}$$

Figure 3.7 shows the efficiency of the three protocols as a function of the bit error rate. The experiment consists of a 1 Mbps channel with one-way propagation delay of 100 msecs and the protocols sending frames of 1500 bytes. In the case of the sliding-window protocols, two variants are included to see the effect of the window, one with $W = 7$ and $W = 127$ frames. The figure presents very expected

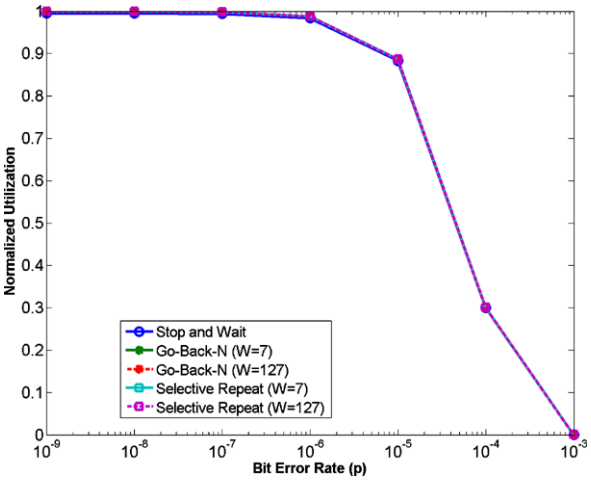


Fig. 3.8 Throughput of ARQ protocols over WSNs.

results. In such a long propagation delay channel, Stop-and-Wait is very inefficient regardless of the BER. The protocol wastes too much channel bandwidth just sending one packet and waiting the round trip time to receive the acknowledgment before sending the following frame. In the case of the sliding-window protocols, it can be seen that both perform similarly in low BER conditions. This is an approximation to the error-free environment, and Equation 3.5 applies. As the channel quality deteriorates, it is expected that Selective Repeat would perform better, which it actually does. The effect of the window size can easily be observed from the figure as well. When W is set to 7, the window size is not big enough to "fill the pipe" with one window worth of frames, and the utilization, in the best case, is 40%. However, this is not the case when W is set to 127, in which 100% utilization is achieved.

The results shown in Figure 3.7, although interesting from the performance of the protocols point of view, are not very interesting for wireless sensor networks. In WSNs, wireless nodes are fairly close to each other and have very low speed links. Further, the frames are also smaller than normal Internet packets. If we redraw Figure 3.7 considering the same frame size but using a channel speed of 250 Kbps and a propagation delay of 100 μsecs, the results, shown in Figure 3.8, say that all protocols perform the same. This is the effect of the low propagation delay combined with the slow speed link and a fairly large frame size.

Given these results and the fact that wireless sensor devices (and most wireless transceivers) work on a half duplex manner, it is not surprising that the Stop-and-Wait protocol is the LLC protocol of choice in most implementations. It is worth mentioning that several Stop-and-Wait variations or hybrid approaches exist in which the Stop-and-Wait protocol is allowed to send an entire window of frames before going to the wait state to wait for the acknowledgment. Two of these variation are described in [81].

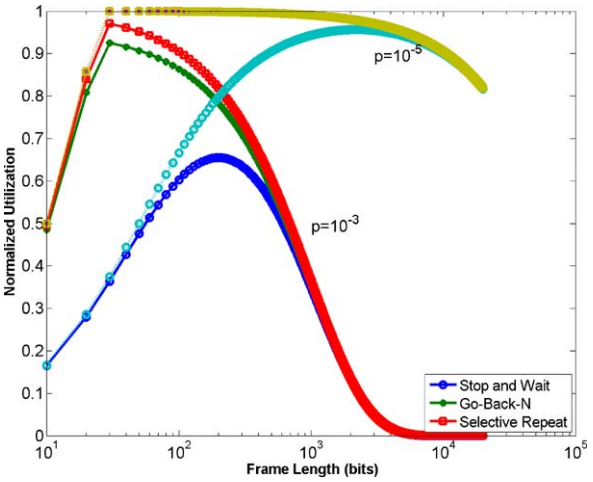


Fig. 3.9 Efficiency of the ARQ protocols as a function of the frame size and channel quality.

3.3.3 Energy Analysis of LLC Protocols

As it has been stated several times, energy conservation is a major concern in WSNs. Therefore, it is important to look into these protocols from the energy consumption point of view. According to the efficiency equations in the last section, the performance of these protocols is a function of several factors, such as the BER, the frame size, the propagation delay, and the link speed. Given a fixed propagation delay and link speed, the channel quality and the frame size are the two most important factors from the energy point of view. When the channel is of a very poor quality, the probability of receiving a frame successfully (regardless of size) is very small and the protocol expends most of its time retransmitting frames. In WSNs retransmissions are undesirable more from the energy consumption point of view than from the efficiency viewpoint, as we know the channel is not well utilized anyway. Also, remember that communications is the most expensive function in wireless sensor devices in terms of energy consumption. Therefore, avoiding or reducing the number of retransmissions is important in WSNs.

Depending on the channel condition, there are different strategies to reduce the energy consumption. One strategy is to find the optimal frame size. If the channel is good, bigger frames make more sense, as they will all go through with less overhead. When the channel conditions are not as good, smaller frames are better, as the probability of having a frame in error is lower with smaller frames. However, smaller frames mean more overhead. Therefore, an optimal frame size exists for each channel condition. The optimal frame size must be studied along with the ARQ protocols, as they have different retransmission strategies. For instance, for every frame in error, Go-Back-N will retransmit more frames than Stop-and-Wait and Selective Repeat.

Figure 3.9 shows the normalized utilization of the ARQ protocols for two different bit error rates while varying the frame length. To draw this particular figure,

a wireless sensor network consisting of a 250 Kbps channel with 100 μsecs propagation delay, and a window of 3 frames for Go-Back-N and Selective Repeat were used. Several observations can be made. First, the utilization of all protocols is low for very small frame sizes. This is due to the combination of two factors, the small frame size, which involves more overhead, and the window size. In this particular case, the latter was the limiting factor. A window size of 3 is not enough to fill the pipe completely. For Stop-and-Wait the window size equal to one is an important limiting factor. Second, as the frame size increases, so does the efficiency of the Stop-and-Wait protocol until an optimal frame size is reached. From that point on, the efficiency decays. Therefore, it is clear that there is a frame size that optimizes the protocol's efficiency. This is the frame size that compromises frame length or number of bits transmitted versus the probability of the frame being in error, or the quality of the channel. These are very important considerations for WSNs as Stop-and-Wait is very widely used. Third, Selective Repeat performs better than Go-Back-N but not in a considerable amount. The difference is only appreciable in higher bit error channels, as Go-Back-N has to repeat more frames. Both protocols can achieve 100% utilization using small frames and large enough windows. Lastly, the efficiency of Selective Repeat and Go-Back-N decreases with the size of the frame. This is expected and it is due to the higher probability of the frame being in error.

Another interesting way of looking at the energy efficiency of the ARQ protocols is to consider the number of retransmissions while varying the quality of the channel and the frame size. This analysis is presented in [68] where the authors use the two-state Markov error model described in Section 2.4 to determine the quality of the channel. According to [68], the mean number of retransmissions is given by:

$$N = (1 - P_e) \sum_{i=1}^{R-1} i P_e^i + R P_e^R \tag{3.9}$$

where R is the maximum number of retransmissions allowed and P_e is the frame error probability, which is given by:

$$P_e = P(Good)\left(1 - (1 - Good_{BER})^L\right) + P(Bad)\left(1 - (1 - BAD_{BER})^L\right) \tag{3.10}$$

where $P(Good)$ and $P(Bad)$ are given by Equations 2.18 and 2.19.

Figure 3.10 shows their results for the Stop-and-Wait protocol with a maximum of $R = 10$ retransmissions for different bit error rates while in the good state and two different frame sizes. The figure says that for a given channel quality a smaller number of retransmissions can be achieved with a smaller frame. For example, when the good state BER is 10^{-4}, the protocol has zero retransmissions if the frame size is equal to 400 bits, but it has to retransmit the frame four times if a 1500-byte frame is used.

Another strategy to reduce the energy consumption is the combined use of ARQ protocols and Forward Error Correction mechanisms. If the channel condition is fairly good, it might be a good idea to implement an ARQ protocol, as frames will be

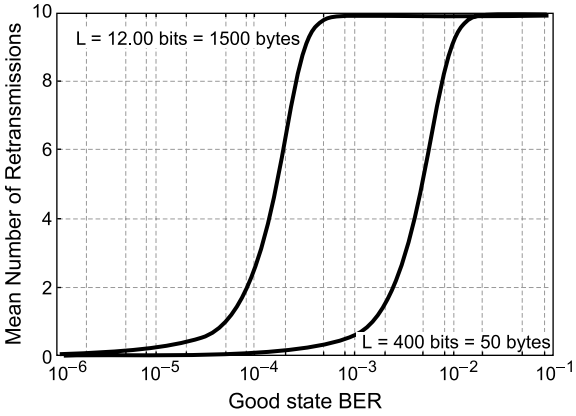


Fig. 3.10 Mean number of retransmissions of the Stop-and-Wait protocol as a function of the channel quality and frame size [68] ©1999 ACM, Inc. Included here by permission.

rarely retransmitted. As the quality of the channel degrades, the ARQ protocol will start retransmitting frames more frequently. At this point, inserting some additional bits to implement a FEC scheme might prove useful. These few extra bits might be enough to reconstruct the original frames at the receiver and will avoid their retransmission. Since it seems like ARQ protocols and FEC mechanisms work better in opposite scenarios, a combined or hybrid approach might be advantageous. Such a combined approach is presented in [68] where the authors go one step further proposing an adaptive hybrid approach that will tune itself according to the time-variant channel conditions. Using the same two-state Markov error model described in Section 2.4, the study investigates the energy consumption per bit comparing

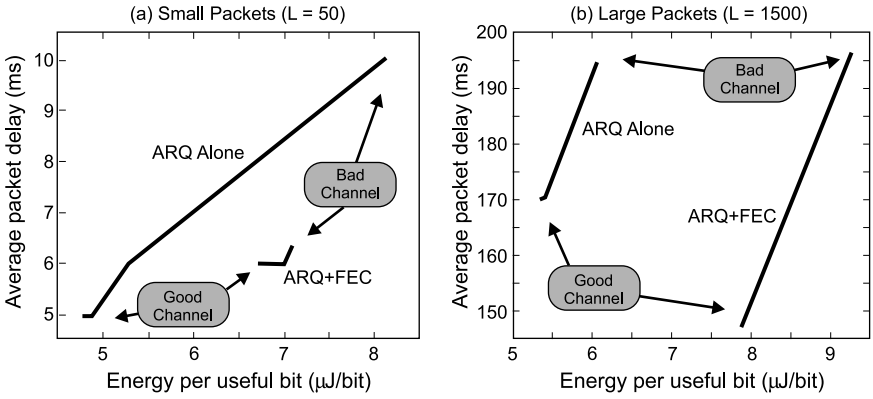


Fig. 3.11 Energy performance of selective repeat ARQ versus a combined approach with selective repeat and FEC as a function of the channel quality and frame size [68] ©1999 ACM, Inc. Included here by permission.

the use of Selective Repeat ARQ alone versus Selective Repeat ARQ and a Reed–Solomon FEC scheme for different frame sizes over Raleigh fading channels.

The results for datagram traffic shown in Figure 3.11 say that for small packets the ARQ strategy alone works best in good channels and worst in low quality channels. The performance of the combined solution is very similar regardless of the channel quality, performing worse than ARQ alone in good channels but considerably better over bad channels. The results are different when big frames are used. The figure shows that for 1500-byte frames, the ARQ strategy alone is better than the combined approach regardless of the channel quality. This is true from the energy consumption point of view only because the figure also shows that the frames might experience longer average delays. However, since delays are not as important for WSNs applications, we can safely say that, for larger packets, the ARQ strategy alone is better.

Chapter 4
The Network Layer

4.1 Introduction

In wireless sensor networks, multi-hop communications is a must given the nodes' limited communication coverage compared with the size of the area of interest. In this case, intermediate nodes must relay packets so they can travel from source to sink. This is the function of the network layer: routing packets from the source node to the destination, normally the sink node, through the wireless multi-hop network. This function is particularly challenging in WSNs considering the computational and energy constraints of the nodes and the network-wide function that it needs to accomplish. This chapter presents the most important routing approaches for WSNs considering small and large-scale networks.

In order to perform the routing function, the network layer utilizes a *routing protocol* and a *routing algorithm*. The routing protocol is the mechanism that the nodes use to exchange network information. This information, which is sent in network layer packets generated by the routing protocol, can be local or global information and may contain one or more variables of information, such as neighbors, paths to destinations, energy available in the nodes, link congestion, average link delay, etc. The exchange of routing information process takes place either periodically, set by the network administrator or designer, or it is triggered based on events, i.e., when the routing information changes.

The routing algorithm is in charge of finding the best possible path from source to destination. It utilizes the routing information collected by the node through the routing protocol to find such a route. The best possible path is chosen according to some predefined optimization criteria related to the network or application needs. For example, it is very common to find the shortest path, as it is the path that utilizes the least amount of network resources and the path that should provide the best delay.

Once the routing protocol and algorithms are run, each node should have a *routing table* that tells the node which is the most appropriate neighbor to forward the packet to in order to reach the destination thorough the best possible path. Routing

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

tables are updated periodically or whenever changes occur in the network in order to have current and valid routing information. One important aspect to consider in any routing function is, therefore, its convergence time, or the time it takes the network to have stable routing tables in all nodes.

In wireless sensor networks the routing function is especially important given the limited availability of resources. Therefore, when designing routing protocols and algorithms, special attention must be given to the following aspects:

- **Simplicity:** The routing algorithm must be simple to implement and must have low computational complexity. Normally, complex optimization functions are avoided in WSNs given the limited computational capabilities of the nodes. Although multiple objective optimization algorithms exist, single metric optimization algorithms are normally used.
- **Energy-efficiency:** The routing protocol must be designed to exchange the minimum amount of routing information. Recall that communication is the most expensive function in terms of energy consumption in WSNs.
- **Scalability:** The routing protocol and algorithm must be scalable with the number of nodes. This is very important for those applications that need a very large number of nodes. In these cases, the complexity of the algorithm and the overhead of the routing protocol should not increase with the number of nodes. Scalability is also associated with the use of local or global routing information. Normally, achieving the routing function with only local information is more scalable than global information, especially in large deployments.

4.2 Routing Protocols for WSNs

Figure 4.1 depicts a convenient way of classifying unicast routing protocols for wireless sensor networks. The classification is based on the knowledge that the nodes have about the network topology after the routing protocol is run. Topology aware protocols have network wide information or global information. Other protocols only provide local, one-hop, or two-hop neighbor information. This is very convenient because this classification relates directly to the size of the wireless sensor network.

4.2.1 Topology Aware Routing Protocols

As the name implies, these protocols exchange network wide or global routing information. These protocols are inherited from routing protocols widely available and implemented in wired networks and wireless mobile ad hoc networks. As such, their applicability in wireless sensor networks is usually restricted to small networks, as these protocols were not designed with the wireless sensor network constraints in mind.

Link state and distance vector protocols are mostly used in wired networks. While in link state routing protocols nodes flood the network with local information,

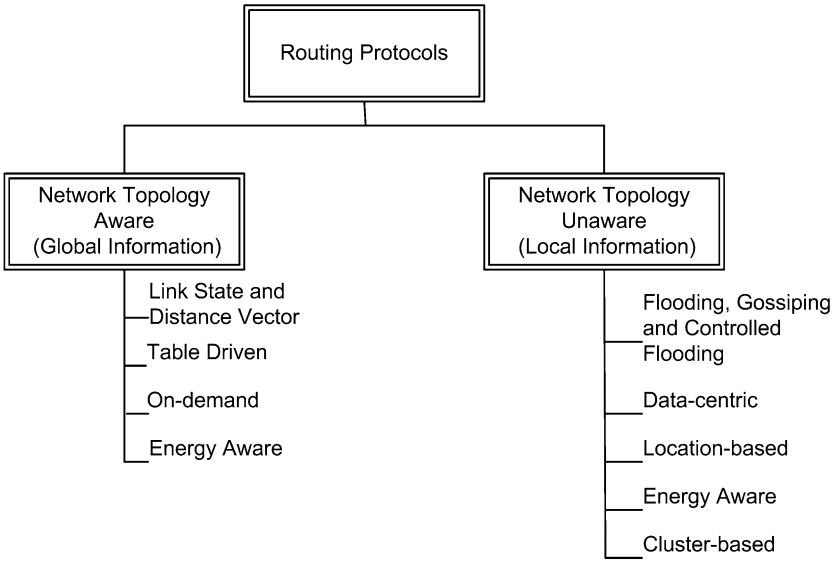


Fig. 4.1 Taxonomy of routing protocols for wireless sensor networks.

distance vector routing protocols exchange network wide information only with local nodes. In both cases, the routing algorithm, which is based on the well-known Dijkstra's or Bellman–Ford algorithms, determines the shortest path from one node to all other nodes in the network considering the number of hops as the optimization metric. These protocols, although scalable for a large number of nodes, are computationally expensive for wireless sensor devices and introduce a large amount of overhead, i.e., they are not energy efficient. Their applicability in WSNs is limited to small networks given that the protocols and algorithms are widely available and proven. The Open Shortest Path First (OSPF) protocol [82] and the Routing Information Protocol (RIP) [48] are the most commonly used link state and distance vector protocols, respectively.

Table driven and on-demand routing protocols are modifications to link state and distance vector protocols designed for wireless mobile ad hoc networks. In particular, they address the problem of frequent link breaks derived from the mobility of the nodes. Nodes in table driven routing protocols or proactive routing protocols send their routing tables as periodic updates to their neighbors or to all nodes in the network. By sending periodic updates the protocol ensures that every node in the network has a picture of the current network topology, and eliminates the presence of stale routes, which is particularly important in highly mobile scenarios. One of the inherent drawbacks of this method is that it increases the routing overhead and reduces the available bandwidth for data packets. The increase in routing overhead is directly proportional to the number of nodes in the network and to the rate of updates. Thus, table driven protocols provide neither good scalability nor perform well in scenarios with little or no node mobility, such as most wireless sensor net-

works. Proactive protocols differ in the number of tables maintained and the method by which changes in the network are broadcast. Destination Sequenced Distance Vector (DSDV) [96] and Wireless Routing Protocol (WRP) [83] are examples of proactive or table driven approaches.

On-demand routing protocols or reactive routing protocols create routes only when required by the source node. When a node requires a route to a destination, it initiates a route discovery process within the network. All nodes in the network participate in this process, and either the destination node, or any other node that has a route to this destination replies back to the source with the complete route information. Each node receiving this information caches the route information and uses it for future packets toward this destination. This approach significantly reduces routing overhead, especially in the case of static networks. There are many ad hoc routing protocols in this category. Dynamic Source Routing (DSR) [16], Ad hoc On-demand Distance Vector (AODV) Routing [95], and Temporally Ordered Routing Algorithm (TORA) [87] are just a few examples.

In general, neither approach is very well suited for wireless sensor networks as energy considerations were not in the designers' minds. Also, neither strategy is scalable when implemented in large networks; not even on-demand protocols, with their reduced overhead in static networks. For example, using source routing in large networks implies very long headers to contain the entire path routing information. In WSNs, this may traduce in a packet where the overhead is many times bigger than the user data, usually a point measure, such as a temperature reading. Also, neither approach is actually simple to implement.

Energy aware routing protocols were the first step toward using power aware routing metrics to increase the network lifetime. Protocols in this category use different types of metrics, such as energy consumed per packet, time to network partition, variance in node power levels, cost per packet, maximum sensor node cost, and the like, with the main goal of extending the network lifetime [113]. Other protocols use battery-related metrics, as they are directly related to the lifetime of the nodes. For example, a protocol can find the route with the maximum sum of available battery capacity, or use a cost function inversely proportional to the available capacity in the nodes. This last metric will make nodes with little energy less likely to participate in the final route. One observation about these protocols is that they normally utilize link state routing protocols to disseminate this power-aware information and use Dijkstra's algorithm or a modified version of it to calculate the routing tables.

4.2.2 Topology Unaware Routing Protocols

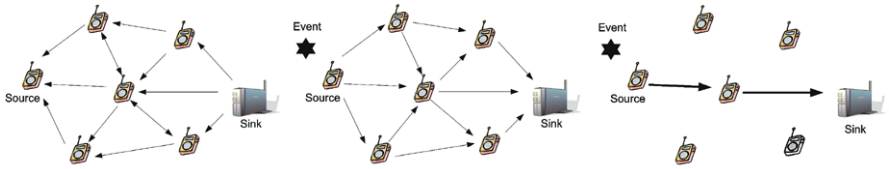
Topology unaware routing protocols are better suited for wireless sensor networks. In fact, most of the protocols designed under this category were completely designed with WSNs in mind. One common characteristic of these protocols is that they use local information to perform the routing function. Flooding and flooding-based protocols, data centric protocols, location-based protocols, energy-aware protocols and cluster-based approaches belong to this category. In the following, the most important protocols will be described.

Flooding, Gossiping, and Controlled Flooding are based on the same design principle: these protocols trade simplicity for reliability and efficiency. Flooding, as the name implies, routes incoming packets to all possible output interfaces. The routing function (protocol and algorithm) could not be simpler than that; however, this simplicity comes with a high cost, very important in WSNs. Packets are forwarded to many more nodes than actually needed, consuming a lot of energy. Although flooding protocols are usually very reliable, it is worth mentioning that reliability is not guaranteed. In order to forward packets, nodes utilize the broadcast capabilities of the MAC layer, which might not implement error control functions to repeat packets when collisions occur. The IEEE 802.11 protocol is one example of a MAC layer protocol that works in this manner.

Gossiping and **controlled flooding** protocols are based on the same idea: reduce the enormous overhead of flooding. One approach is to utilize a **randomized forwarding** approach, like the one in [46], in which nodes forward incoming packets with a particular probability. This approach has been shown to spread the information to almost all nodes in the network if this probability is around 65 to 75%. Lower values of this probability have the effect of quickly killing the message, reaching a very small number of nodes. **Rumor routing** [15] is another strategy to reduce the overhead of flooding mechanisms. Rumor routing is based on the idea that two random lines in a square have a relatively high probability to intersect. As such, the protocol sends agents to embed routing and event information in all the nodes that they visit. When one of the nodes in the network detects an important event, another agent is generated that goes through the network until it intersects with one of the nodes that has been previously embedded with routing and event information. If the event information matches, the node utilizes the routing information to send the event to the appropriate node, normally the sink. If five event agents are sent instead of one, the intersect probability is shown to increase from 69 to 99.7%.

Data centric routing is based on the idea that routing, storage, and querying mechanisms can be performed more efficiently if communication is based on application data instead of the traditional IP global identifiers. This data centric approach to routing provides additional energy savings derived from the communication overhead of binding identifiers that are no longer needed, and facilitates in-network processing with additional savings derived from data aggregation and compression. Two important examples of data centric routing protocols are **Sensor Protocols for Information via Negotiation (SPIN)** [66] and **Directed Diffusion** [56].

SPIN [66] is designed to address the deficiencies of classic flooding. SPIN uses negotiation and resource adaptation, as each node disseminates information to every other node in the network considering them to be potential base stations. SPIN is based on the idea that sensor nodes operate more efficiently and conserve energy by sending data that describe the sensor data instead of sending all the data. SPIN uses *data advertisement messages (ADV)*, *request for data messages (REQ)* and *data packets/messages (DATA)*. Initially, a sensor node broadcasts an ADV message containing a description of the data it has. The neighbor nodes who are interested in the data send a REQ message to the first sensor node. The sensor node holding the data then sends the DATA message to the nodes from which it received the



(a) The first phase: the flooding of interest.

(b) The second phase: setup of gradients.

(c) The third phase: the pruning phase.

Fig. 4.2 The Directed Diffusion protocol.

REQ message. The neighbor nodes then repeat this process. Finally, all nodes in the network interested in the data will have a copy of the data. In SPIN topological changes are localized since each node needs to know only its single-hop neighbors.

The Directed Diffusion protocol [56] works in three phases. In the first phase, the base station floods a task description, which is also known as interest, to all the sensor nodes in the network. The naming of task descriptors is done by assigning attribute-value pairs that describe the task. During the second phase each sensor node saves the interest in its memory and creates a gradient towards the nodes from which it received the interest. When a sensor node has sensor data that matches the interest, it starts sending the data to all the neighbors it has gradients to. This process repeats from node to node using the gradients until the sink node starts receiving the data. The last phase consists of a pruning mechanism. Upon receiving data from multiple neighbors, the sink starts reinforcing one particular neighbor or k neighbors, sending packets towards the source increasing the value of the gradients. This phase eliminates many routes and therefore reduces the amount of overhead considerably. Figure 4.2 depicts the three phases of the protocol.

Location-based routing has recently emerged as one important approach to address the scalability and energy efficiency concerns for data dissemination in wireless sensor networks. In location-based routing, packets are routed based on the location of a set of candidate neighbors and the location of the final destination. As a result, nodes do not need to make complex computations to find the next hop, since routing decisions are made based on local information. Moreover, because of the locality, nodes do not need to maintain large data structures. Finally, location-based routing reduces the communication overhead substantially because routing table advertisements, like those in traditional routing protocols, are not needed. While these are important features, very well in line with the constraints and characteristics of wireless sensor networks, it is worth noticing that nodes need to know their locations and the location of the destination, which are not always easy or efficient to obtain. Fortunately, the static nature of most wireless sensor networks makes it possible to establish a virtual relative coordinate system that the nodes can use to know their locations. Examples of this virtual system and a location service mechanism can be found in [146, 78]. This greatly reduces the extra cost and overhead of GPS-based and other localization mechanisms needed in mobile networks.

Figure 4.3 depicts several alternatives widely known for making local routing decisions based on the location of the source's neighbors and the location of the des-

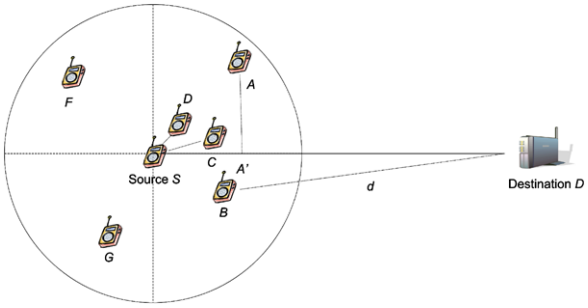


Fig. 4.3 Location-based routing alternatives.

tination. The most forward within radius strategy forwards the packet to the neighbor that is closer to the destination with respect to the source–destination direct line (node *A* in Figure 4.3). The Euclidean distance can be used to forward the packet to the node with the smallest distance to the destination (node *B* in Figure 4.3). This strategy, although very simple, ignores the topology of the network and sometimes is not able to find the shortest possible path. Another routing strategy is to forward the packet to the nodes that least deviates from the source–destination line (node *C* in Figure 4.3). This strategy, also known as directional routing, is utilized in the **Distance Routing Effect Algorithm for Mobility (DREAM) protocol** [7], which will be explained later. Finally, there is also the strategy that forwards the packet to the closest node to the source that still advances the packet towards the destination (node *D* in Figure 4.3).

Normally, these schemes only consider the nodes in the backward direction (nodes *F* and *G* in Figure 4.3) when no nodes are available in the forward direction. When no nodes are available in the forward direction, the protocols need of additional mechanisms to avoid routing loops. One of those mechanisms is based on the use of memory, so the nodes know which nodes have already participated in the routing of the packet, and stop the packet forwarding process when the best neighbor to send the packet to is the one from which the packet came from. In [118], the authors show that the most forward within radius strategy is loop free but the directional strategy is not. Another problem of these routing strategies is their inability to deal with dead ends, void areas, and the end of the area of interest. In [13, 14], the authors introduce the idea of **face routing** using the **right hand rule** or the **left hand rule** to guarantee packet delivery and routing over void areas. One restriction of face routing is that it only works on geometric planar graphs, or graphs in which there is no intersection between any two edges. An example of a routing protocol that combines these two mechanisms is the **Greedy Perimeter Stateless Routing (GPSR) protocol** [62]. GPSR utilizes the most forward within radius strategy with face routing.

Another important location-based routing protocol is Distance Routing Effect Algorithm for Mobility (DREAM) [7], which was designed for wireless mobile ad hoc networks. In DREAM each mobile node maintains a location table for all other

nodes in the network. Each location table consists of the coordinates of the source node based on some reference system, the source node's speed, and the time the location packet was transmitted by the source node. Each mobile node transmits location packets to nearby mobile nodes in the network at a given frequency and to faraway mobile nodes in the ad hoc network at a lower frequency. This is to reflect the distance effect, or the fact that distant nodes appear to be moving more slowly than closer nodes. When a node S needs to send information to node D, it calculates a circle around the most recent location information of D using Equation 4.1,

$$R = V_{max} \times (t_1 - t_0) \tag{4.1}$$

where R is the radius of the circle, V_{max} is the known maximum speed of node D, t_1 is the current time and t_0 is the timestamp for this node's information from the location table. Then node S defines a forwarding zone whose vertex is at S and whose sides are tangent to the circle around D. S then sends a data packet to all its neighbors in the forwarding zone. The neighbor nodes in turn calculate their own forwarding zones to D and repeat the process. Mobile node D on receiving the data packet sends an acknowledgment packet (ACK) to S. The ACK packet is sent to S in the same manner as the data packet was send to D. If node S does not receive an ACK packet it uses a recovery procedure.

Energy aware routing protocols consider energy in the routing function to extend the network lifetime and distribute the energy consumption more evenly among the nodes. The **Energy Aware Routing (EAR) for Low Energy Ad-hoc Sensor Networks protocol** [110] is an example of a protocol in this category. EAR is based on the idea of not using the minimum energy path all the time. Instead, the protocol route packet using sub-optimal paths using a probability function which depends on the energy consumption of each path. In EAR the nodes build their routing tables using the following procedure: First, the destination node starts the procedure flooding the network with an initial cost equal to $Cost(N_d) = 0$. Receiving nodes compute the total cost of the path so far using Equation 4.2

$$C_{N_j,N_i} = Cost(N_i) + Metric(N_j, N_i) \tag{4.2}$$

where $Metric(N_j, N_i)$ considers the cost of transmitting and receiving the packet from node i to j, respectively and the residual energy at node i.

Then, each node builds its routing table assigning a probability inversely proportional to the cost to each of its neighbors, as follows:

$$P_{N_j,N_i} = \frac{1/C_{N_j,N_i}}{\sum_{k \text{ neighbors}} 1/C_{N_j,N_k}} \tag{4.3}$$

Finally, node N_j calculates its own cost, given by Equation 4.4, which is the information that node N_j sends to node N_i, and the process repeats.

$$Cost(N_j) = \sum_{i \text{ neighbors}} P_{N_j,N_i} C_{N_j,N_i} \tag{4.4}$$

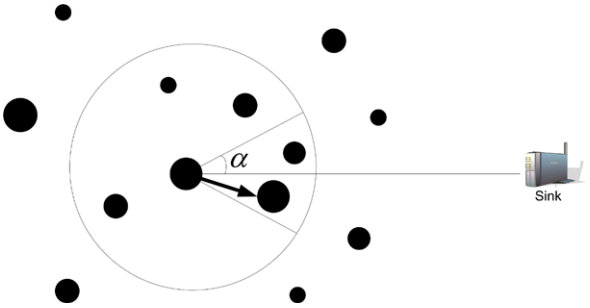


Fig. 4.4 SELAR: a location-based energy aware routing protocol.

Upon receiving a packet, node N_j forwards the packet to the neighbor with the highest probability in the routing table. In this manner, packets are forwarded using the nodes that consume less energy and have more residual energy. Also, since the probabilities change over time according to the residual energy in the nodes, the energy is more evenly consumed and the network lifetime is extended.

Another energy aware protocol is the **Scalable Energy-efficient Location-Aided Routing (SELAR) protocol** [76, 77]. SELAR is a location-based routing protocol that borrows several of its mechanisms from DREAM but includes the residual energy of the nodes to make the forwarding decision. SELAR implements the *most forward with more residual energy within angle* forwarding strategy. SELAR establishes an initial angle $\alpha = 15°$ in the direction of the destination and forwards the packet to the farthest neighbor with more residual energy. The angle α is increased by 15° increments if no neighbors are found. One drawback of SELAR is that it can't guarantee packet delivery and routing over void areas, as it does not implement face routing or a similar solution. Figure 4.4 depicts the idea of the protocol in which the size of the node is proportional to its residual energy.

Cluster-based routing is another approach to reduce the routing complexity and therefore make routing more scalable and reduce energy consumption. In this category the **Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol** [47] is the seminal work in wireless sensor networks. LEACH combines several important ideas with the goal of extending the network lifetime and consuming the network energy evenly.

- LEACH is a distributed algorithm by which nodes group together in clusters and select themselves as clusterheads. Since clusterheads perform more energy consuming functions than the rest of the nodes, LEACH not only changes them periodically but it also makes sure that clusterheads are selected in a fair and uniform manner, so energy is consumed in an even manner. The algorithm makes sure that the probability of a node to become a clusterhead given that it has already been clusterhead before is very low compared with the rest of the nodes.
- Intracluster communication is performed using TDMA at the MAC layer. This allows nodes to go to sleep when there are no transmission for them during the incoming schedule, saving additional energy.

- Nodes only communicate with their respective clusterhead, which is near by; therefore, nodes don't need to spend much energy to send their data.
- LEACH performs data aggregation in the clusterheads, a very convenient place given that they receive all the data from the nodes in their clusters. This saves additional energy.
- The clusterhead transmits the data directly to the sink using CDMA. CDMA avoids collisions and interference when several clusterheads are transmitting to the sink at the same time. Because of the direct transmission, there is no routing protocol and overhead associated with sending the data.

Although LEACH has been shown to achieve its objectives, one major drawback is that the protocol is not scalable in terms of area of coverage. The fact that clusterheads have to transmit directly to the sink limits the area of coverage to the area of coverage of the nodes. This scalability problem is addressed in HEED [142], a cluster-based routing protocol similar to LEACH. In HEED, clusterheads form an overlay network that perform the routing function routing packets from cluster to cluster until they reach the sink.

Chapter 5
The Transport Layer

5.1 Introduction

In the communication protocol stack, the transport layer is in charge of the transmission of segments on an end-to-end basis. The transport layer, which interfaces with the application layer, currently provides two types of services: a reliable and an unreliable service.

The User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP) are the transport layer protocols of choice in the current Internet. It is very well known that UDP provides neither message delivery reliability nor includes a mechanism for flow and congestion control. As a result, UDP has been utilized by those applications that need a continuous delivery of segments and can tolerate some losses, such as streaming applications, like voice over IP, or video applications. TCP, at the other side of the spectrum, provides a very reliable service and includes mechanisms for flow and congestion control. TCP has been the protocol of choice for data-oriented applications that do not send data continuously but cannot tolerate packet losses, such as transactions over the web, file transfers, telnet sessions, and the like.

Unfortunately, wireless sensor networks have different constraints compared to wired and even other wireless networks. Also, wireless sensor network applications differ from Internet-like applications in many aspects. As a result, these two factors make current transport layer protocols, such as UDP and TCP, inappropriate for wireless sensor networks. This chapter first presents a general view of the functionality of the transport layer. Then, wireless sensor network applications and their characteristics are described with the goal of identifying which of those transport layer functions are needed to support each type of application. Finally, the chapter describes some of the most important transport layer protocols available in the wireless sensor networks literature.

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

5.2 Transport Layer Functions

The transport layer of the communication protocol stack is in charge of flow and congestion control and error control functions on an end-to-end basis. Different transport layer protocols are then designed including both functions, only one of them or even none of them. For example, TCP includes flow and congestion control and error control while UDP includes none.

Flow control is the mechanism that guarantees that the sender will not overwhelm the receiver with more data than it can receive and process. Congestion control, on the other hand, is the mechanism that guarantees that the network is not overwhelmed by more traffic that it can handle. As a result, it is clear that flow control is an end system issue while **congestion control** is a network issue. In this chapter, we will only consider the congestion control issue, as it is more relevant than flow control in wireless sensor networks.

Congestion control has three main components: congestion detection, congestion notification, and congestion reaction. **Congestion detection** is the mechanism used by the transport layer protocol to detect congestion. This mechanism can act in a proactive or a reactive manner. A proactive mechanism can be used to prevent or avoid congestion while a reactive mechanism usually acts once congestion has taken place. Mechanisms widely used to detect congestion are repeated acknowledgments, sender timeouts, and buffer size measurements. Once congestion has been detected, a notification needs to be conveyed to the sender so it can act upon it. **Congestion notification** can be explicit, by sending a message to the source indicating the location and level of the congestion, or implicit by means of measured end-to-end delays or acknowledgments. Finally, once the source knows that the network is congested, it has to react accordingly. This is the **congestion reaction** component and there are many different mechanisms to change the source's transmission rate in response to the congestion notification received. The most widely known congestion reaction is the Additive Increase Multiplicative Decrease (AIMD) strategy used by the TCP protocol, which increases the transmission rate in an additive manner if transmissions are successful, and reduces the transmission rate in a multiplicative manner on error conditions.

Error control is the function in charge of detecting and recovering from missing or corrupted segments. Error control consists of **error detection** and **error correction** mechanisms. While missing packets are usually detected using sequence numbers, corrupted packets are detected by means of error checking mechanisms. Error correction mechanisms are usually based on end-to-end retransmissions.

5.3 Wireless Sensor Network Applications

In order to design appropriate transport layer protocols for wireless sensor networks it is important to know the type, characteristics, and requirements of the most common applications utilized over these networks. Figure 5.1 shows the domain space for wireless sensor network applications, based on whether the applications send one or multiple packets, and the level of reliability that they need.

	Low Reliability	High Reliability
Single Packet	Event Reporting	Aggregated data
Multiple Packets	Streaming Applications	Re-tasking

Fig. 5.1 Application domain space in wireless sensor networks.

5.3.1 Single Packet–Low Reliability Applications

Many sensor networks are densely deployed to measure and report a particular variable to the base station (sink) over time. In these applications, each sensor node transmits its measurement to the sink whenever a threshold is reached. For instance, the network will report the field temperature if it goes beyond 100 °F. As many sensors close to each other may experience a similar temperature, they may all be reporting similar readings to the base station. These types of applications don't need a high level of reliability, as it is expected that at least one of the packets will eventually arrive at the sink. These applications look for *event reliability* more than *packet reliability*.

From the transport layer protocol point of view, these applications may be well served by a simple and rather unreliable protocol, or by a more reliable data link layer protocol. [49] shows that while an end-to-end retransmission strategy is better in good quality channels, hop-by-hop strategies are more energy efficient in low quality channels. A similar conclusion is also reached when considering the number of hops n in the communication path. While end-to-end retransmissions are more energy efficient when n is small for a fixed and rather high bit error rate channel, hop-by-hop strategies are better when n is large.

One transport layer protocol for wireless sensor networks under this category is the **Hop-by-Hop Reliability (HHR) protocol** described in [29]. Given a desired level of reliability and a locally estimated packet loss rate, HHR calculates the number of times a node has to send the same packet to one of its neighbors so that the desired level of reliability is achieved at the sink over n hops. HHR also aims at being energy efficient by not requiring the receivers to send any acknowledgment. The authors show that the overhead incurred in repeating the packets is less than the one used by an end-to-end retransmission strategy. The authors propose three protocol variants called Hop-by-Hop Reliability with Acknowledgment (HHRA), Hop-by-Hop Broadcast (HHB), and Hop-by-Hop Broadcast with Acknowledgment (HHBA). HHRA is similar to HHR but the receiver sends acknowledgments and the sender stops repeating packets as soon as the first acknowledgment is received. HHB is similar to HHR but instead of sending the packets to one neighbor, it broadcast n copies of the packet. Finally, HHBA is like HHB but with acknowledgments, as in HHRA. The authors conclude that under good channel conditions and low

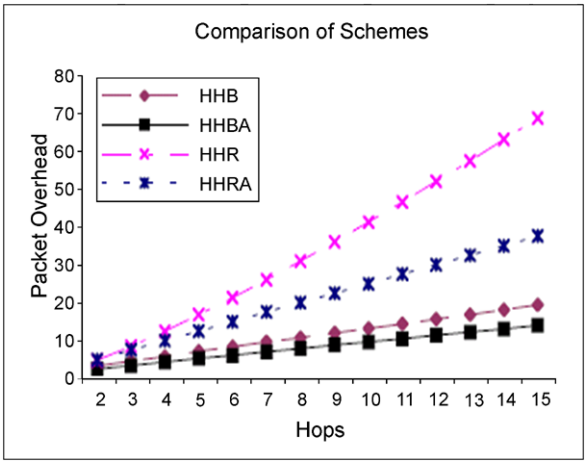


Fig. 5.2 Packet overhead of the HHB, HHBA, HHR and HHRA protocols in a scenario with 50% packet loss rate and 70% of desired reliability level [29] ©2003 ACM, Inc. Included here by permission.

number of hops the HHR and HHB protocols perform better than their acknowledgment counterparts. However, the versions with acknowledgments reduce the overhead significantly in opposite scenarios. Figure 5.2 compares the overhead of the four protocols described before.

Another scheme in this category is the **ReInForM protocol** introduced in [30]. ReInForM is similar to the HHR protocols in the sense that redundant packets are sent to achieve a desired level of end-to-end reliability. Instead of sending redundant packets to the same neighbor, as in HHR, ReInForM sends multiple copies through different paths from source to sink (see Figure 5.3). Packets carry network condition information to make forwarding decisions at each node. In this manner, the level of reliability changes dynamically based on network conditions and the efficiency of the protocol is improved. One drawback of this scheme is that nodes need to know the number of hops n_s to reach the sink in order to calculate the appropriate level of reliability r_s, and also need to calculate the local packet error rate p. Equation 5.1 is used by ReInForM to calculate the number of paths P.

$$P = \frac{log(1 - r_s)}{log(1 - (1 - p)^{n_s})} \tag{5.1}$$

Finally, the **Event-to-Sink Reliable Transport (ESRT) protocol** described in [103] is another important protocol that aims at providing reliable event detection in wireless sensor networks while being energy efficient. ESRT works based on the idea of adjusting the reporting rate at the source in order to reach the target event reliability level. Most of the functionality of ESRT is included in the sink in order to keep sensor nodes simple. The approach taken by ESRT functions on an end-to-end basis, like any transport layer protocol. Although end-to-end reliability in wireless

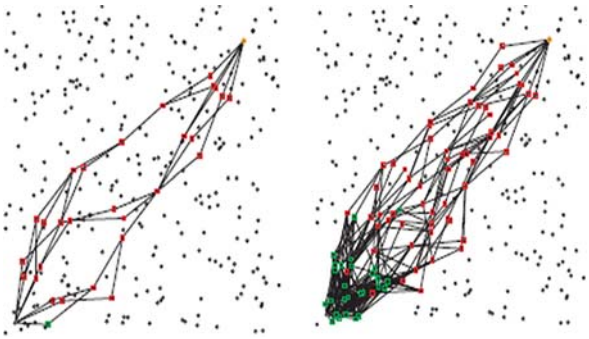


Fig. 5.3 Illustration of the ReInForM mechanism when sending 10 packets over zero error channels (left), and the same 10 packets over channels with 30% error rates. Reproduced from [30] © 2003 IEEE.

sensor networks is most of the time less energy efficient than hop-by-hop reliability, end-to-end schemes can adjust better to application requirements.

5.3.2 Single Packet–High Reliability Applications

There are several cases in which wireless sensor network applications need a high level of reliability. One of the most important examples is the network that uses in-network processing aggregating data. For example, in a cluster-based topology, it is very useful to ask the clusterhead to aggregate similar data and send only one packet summarizing the data it received from several of its cluster members. The aggregated data can be an average of the temperature measured by the different nodes, the minimum and maximum readings, etc. The issue is that now only one packet is sent to the sink instead of many redundant ones like in the single packet–low reliability applications explained before. Therefore, these applications need a high level of packet reliability. Protocols such as HHR and its variants, ReInForM, and ESRT could be used here as well, as they can provide a high level of reliability. Another option in this category is to use TCP. However, TCP suffers from several drawbacks that limit its applicability in wireless sensor networks (more on this later).

5.3.3 Multiple Packet–Low Reliability Applications

These are applications in which sensor nodes send streaming data to the sink, such as voice or video. These applications are more concerned about timing issues, to be able to play the stream at the receiving end, than packet loss rates, as they can absorb some losses. Streaming applications are usually very energy consuming because they send many packets. Normally, wireless sensor networks are not utilized for these types of applications because of the resource constraints. However, it is expected that more powerful devices will be soon available that will make wireless sensor networks more adequate for streaming applications. In this case, the entire communication protocol stack might need to be revisited again to provide delay and

bandwidth guarantees and provide other types of services. The reader is referred to [2] for a survey on wireless multimedia sensor networks. Another option here is to use the UDP protocol.

5.3.4 Multiple Packet–High Reliability Applications

All applications described so far are of the many-to-one type, in which nodes send data to the sink. There are other types of applications in wireless sensor networks which are of the one-to-many type that send multiple packets and need very reliable service. One example of these applications are re-tasking applications. Re-tasking applications send new code from the sink to the nodes to change the functionality of the network. The code could go from a few packets to change the measurement threshold of an application reporting the temperature in a particular field to many packets to change the complete behavior of the network. From the transport layer point of view, re-tasking applications need a very reliable service, so all nodes receive the original code without errors for the network to function as expected.

One important protocol in this category is the **Pump Slowly, Fetch Quickly (PSFQ) protocol** presented in [123]. PSFQ design is based on hop-by-hop retransmissions based on the fact that wireless sensor networks are multi-hop networks with very low quality channels. In this scenario, errors accumulate in an exponential manner over multi-hops and end-to-end retransmissions are neither effective nor energy efficient. PSFQ also saves additional energy by using a negative acknowledgment reporting strategy. Since applications using PSFQ send multiple packets, a NACK-based loss notification strategy can be utilized and therefore the energy related to sending many positive acknowledgments can be saved.

PSFQ consists of three operations, namely pump, fetch, and report operations. The pump operation is originated by the sink, which slowly and periodically broadcast all packets to its neighbors. The fetch operation is triggered by the receiving node every time it detects a missing packet. This node immediately (quickly) sends a NACK to the sender asking for the missing packet. The sending node is supposed to cache the packets until they are correctly acknowledged. The report operation is also initiated by the sink and is meant to provide the sensor nodes with data delivery information. This mechanisms is designed to avoid the message feedback implosion created by reporting data generated by all nodes to the sink. The reporting mechanism travels from the furthest node to the sink on a hop-by-hop basis. Each node along the path appends its reporting information in a single packet that aggregates all reporting data.

In [117], the authors study the issue of where in the wireless sensor network's protocol stack it is better to implement reliability – at the MAC layer or at the transport layer. In the same study, the authors propose the **Reliable Multi-Segment Transport (RMST) protocol** [117], a protocol designed to provide data transfer reliability to directed diffusion [56] when sending blocks of data from nodes to sink. RMST comes in two variants. The first one implements a hop-by-hop NACK-based notification and recovery strategy. This is a transport layer hop-by-hop strategy. Here, the protocol caches segments in intermediate nodes and propagates the

NACK segment along the route if the missing segment is not found. The second variant works on an end-to-end basis caching packets at the sink and propagating the NACK signal all the way to the sink. This is a pure end-to-end strategy. The evaluation of RMST is performed using three MAC layers under three different error conditions. The first MAC layer utilizes broadcast as its communication mechanisms, and it is unreliable, i.e., it neither repeats packets nor use any type of control packets. The second MAC layer uses the Stop-and-Wait protocol with RTS–CTS–ACK packets via unicast transmissions. The last one is called Selective ARQ and is a combination of the last two. It unicast data packets including its related control packets and broadcast higher layer control packets, such as routing advertisements.

The performance evaluation of RMST provided the following interesting conclusions. First, under low quality channel conditions (10% BER or more at the physical channel), some sort of hop-by-hop error notification and recovery mechanism is needed; pure transport layer mechanisms are not able to guarantee packet delivery. On the other hand, under very good channel conditions, the end-to-end transport layer mechanism with the unreliable MAC protocol is the best combination. Second, if the reliable MAC protocol is used, transport layer hop-by-hop recovery does not improve the reliability much. Third, it is better to provide transport layer hop-by-hop reliability than MAC layer reliability.

5.4 Congestion Control in Wireless Sensor Networks

So far, we have only considered the reliability needs of the applications and the protocols supporting them. However, not much has been said about congestion control. Although it is easy to foresee that many of the applications already described will not produce an important level of network congestion, multiple packet applications may produce some, especially close to the sink node, since all packets converge there. Therefore, network congestion is an important issue in some scenarios, and as such, it has been considered by the research community.

Three relevant protocols consider congestion in their design. First, it is the **COngestion Detection and Avoidance (CODA) protocol** described in [124], which regulates multiple sources associated with the same event in case of congestion. CODA utilizes the queue length at intermediate nodes as the congestion detection mechanism and a local back-pressure mechanism for congestion notification. The problem with CODA is that these mechanisms are not as fast and energy efficient as desired. The second protocol is ESRT described before. ESRT adjusts the sensors packet generation rate based on the number of packets that have arrived at the sink. ESRT adjusts the transmission rate without causing congestion. Finally, the **STCP protocol** presented in [59] is designed to support multiple applications in the same network with different levels of reliability and congestion detection and avoidance. As in ESRT, the majority of the functionality is implemented in the sink to avoid having complex sensor nodes. STCP also uses the queue length as the congestion detection mechanism, utilizes an implicit congestion notification strategy, and uses an AIMD-like end-to-end congestion reaction rate adjustment.

5.5 The Use of TCP and UDP in Wireless Sensor Networks

As said in the Introduction of the chapter, UDP and TCP are the transport layer protocols of choice in the current Internet; however, it was also stated that they are not well-suited for wireless sensor networks without being specific as to why this is so. The list that follows addresses this aspect.

- **Reliability:** TCP and UDP provide completely opposite services as regards to reliability. Unfortunately, it is clear from the applications described earlier that wireless sensor network applications are not binary, i.e., needing very reliable or very unreliable service, rather they need different levels of reliability. The problem is that, even though UDP may work fine in those cases where applications don't need reliability at all, TCP is not a good choice for highly reliable cases, as it may not be energy-efficient.
- **Energy efficiency:** TCP was not designed with energy constraints in mind. In fact, TCP is not an energy efficient protocol. For example, TCP connection setup might not be needed by some WSN applications, such as those sending a single packet. Data reporting applications sending readings periodically will consume more energy establishing and tearing down the end-to-end connections than sending the data. Imagine a three-way handshake versus a single packet with a temperature reading. Also, TCP has very well-known problems running over wireless networks. These problems might be even worse over WSNs given the reliability of the nodes and the reduced transmission power, and exacerbated in large-scale WSNs. End-to-end retransmissions may end up spending a lot more energy than hop-by-hop retransmissions.
- **Simplicity:** UDP is a very simple protocol but TCP is not. Although light versions of TCP have been proposed [79, 32] for WSNs, TCP is a very heavy protocol in terms of lines of code and memory consumption to be implemented in wireless sensor devices. Protocols for WSNs must be lightweight.
- **Flow and congestion control:** TCP's flow and congestion control is unfair to those nodes with longer round-trip times, i.e., nodes closer to the sink will get better performance than those far away from it. Another aspect is that many applications for WSNs send very small periodic or event-based packets to a powerful sink node. Therefore, the network is underutilized most of the time and there are no concerns about overflowing the receiver, i.e., flow and congestion control are not needed most of the time. In other applications, the sink is actually overwhelmed since all upstream traffic converges there; therefore, some form of flow and congestion control might be needed.
- **Aggregation:** TCP works on an end-to-end fashion and therefore does not allow for in-network processing and aggregation.

In summary, it can be said that the degree of reliability and flow and congestion control to be provided by the transport layer protocol highly depend on the application and the network topology. In addition, the protocols must be simple and energy efficient. Given the poor reliability of wireless sensor networks and the worse effect on large scale networks, the old debate of end-to-end versus hop-by-hop reliability needs to be considered in the design of new protocols. Similarly, the concept of

cross-layer design by which the traditional layered approach is no longer respected and layers help each other directly in the achievement of better system performance is also worth considering in the design of transport layer protocols, as it can be a major source of energy savings in WSNs.

Chapter 6
Topology Control

6.1 Introduction

Wireless sensor networks continue to be a very popular technology to monitor and act upon events in dangerous or risky places for humans. As such, WSNs have been installed in places such as chemical plants, to monitor poisonous gases; water plants, rivers, lakes and the like, to assess the level and quality of the water; endangered species, to monitor their travel patterns and behaviors; buildings, to monitor the quality of the air and make them more energy-efficient; the military, to detect intruders; and many more applications.

Although WSNs have evolved in many aspects, they continue to be networks with constrained resources in terms of energy, computing power, memory, and communications capabilities. Of these constraints, energy consumption is of paramount importance, which is demonstrated by the large number of algorithms, techniques, and protocols that have been developed to save energy, and thereby extend the lifetime of the network.

This chapter formally introduces the concept of *Topology Control (TC)*, as it is one of the most important techniques utilized to reduce energy consumption in wireless sensor networks. The chapter explains in more detail the motivations behind these algorithms and includes a discussion about where to locate the topology control algorithm within the communication protocol stack and what are the necessary or possible interactions with the adjacent layers. Finally, it includes the taxonomy of topology control that will serve as the basis for the following chapters.

In this book topology control encompasses a broader definition than the definition normally found anywhere else. We define topology control as an iterative process, as shown in Figure 6.1. First, there is an initialization phase common to all wireless sensor network deployments. In this phase, nodes discover themselves and use their maximum transmission power to build the initial topology. After this initialization phase, the second phase builds a new (reduced) topology. We call this phase **Topology Construction**. This new topology will run for certain amount of time, as the participating sensors will consume their energy over time. Therefore, as

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

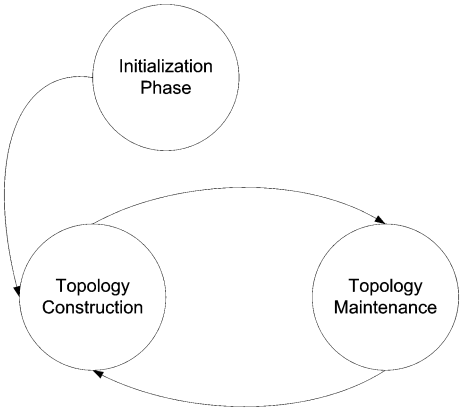


Fig. 6.1 The topology control cycle.

soon as the topology construction phase establishes the reduced network, the **Topology Maintenance** phase must start working. During this phase, a new algorithm must be in place to monitor the status of the reduced topology and trigger a new topology construction phase when appropriate. Over the lifetime of the network, we expect this cycle to repeat many times until the energy of the network is depleted. There are many different algorithms that can be used in the topology construction and maintenance phases. These algorithms are what the second part of this book is all about.

6.2 Motivations for Topology Control

The first question we need to answer is: Why do we want to perform such a complex iterative process? This section briefly examines the main motivations for doing topology control in wireless sensor networks. Also, at the end, it summarizes the major concerns and challenges in applying this iterative process.

6.2.1 Energy Conservation

The main motivation behind the topology construction phase is to build a reduced topology that will save energy and preserve important network characteristics, such as connectivity and coverage. Topology construction, as it will be explained in later chapters, can be exercised by either reducing the transmission power of the nodes or by turning unnecessary nodes off.

Reducing the transmission power of the nodes has the immediate effect of eliminating certain direct links and forcing packets to go through multiple hops. Therefore, the energy savings obtained by using this method are not only related to the fact that the nodes now transmit at lower power levels, but are also related to a series of multi-hop communications instead of longer and direct links. These energy savings can be easily explained using Figure 6.2 as an example, and the log propagation model described in Chapter 2.

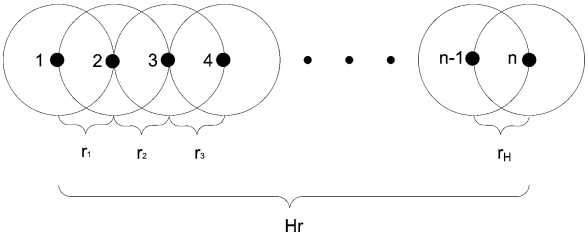


Fig. 6.2 Multi-hop versus single hop power advantage.

The figure shows a linear network with H hops where nodes are equally spaced by distance r. Therefore, the overall distance is given by Hr. Then, in order for node 1's signal to reach node n, it will need to use a transmit power level given by:

$$P_{tx} = P_{rx}(Hr) \times (Hr)^2 \tag{6.1}$$

assuming a path loss exponent $\alpha = 2$. On the other hand, the power needed by each node to reach its adjacent neighbor is given by:

$$P_{tx} = P_{rx}(r) \times r^2 \tag{6.2}$$

Therefore, the energy savings derived from using multi-hop communications are given by:

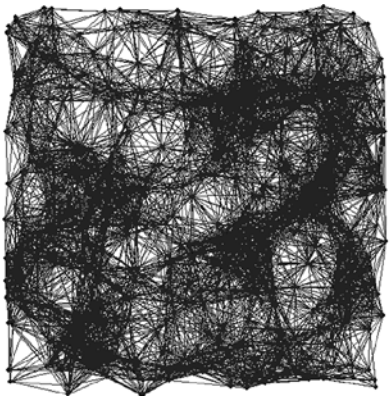
$$P_{savings} = \frac{P_{rx}(Hr) \times (Hr)^2}{H \times P_{rx}(r) \times r^2} = H \tag{6.3}$$

which means that the larger H the larger the power savings. However, this is a very simplistic analysis because it ignores the power consumed by the nodes receiving the signal. In the direct link case, only the n^{th} node receives the signal, but in the multi-hop case, $(n-1)$ nodes spend energy receiving the signal. Therefore, caution is advised in this regard. Also, other aspects need to be considered, such as end-to-end delay and increased probability of packet loss, that call for a balance design that optimizes these metrics.

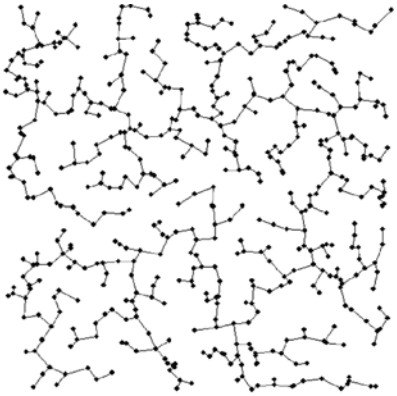
Turning unnecessary nodes off also have the immediate effect of eliminating certain direct links and forcing packets to go through multiple hops. However, further energy savings can be achieved since nodes not part of the active topology can be completely turned off. Nodes take active and inactive turns in order to spend their energy in a fairly manner and increase the network lifetime.

6.2.2 Collision Avoidance

The topology construction phase also has the collateral effect of reducing packet collisions at the data link layer, reducing the number of retransmissions, and consequently, additional communication costs. In densely deployed WSNs where all nodes transmit at their maximum power, the *Maximum Power Graph (MaxPowerGraph)* shows that each sensor has many neighbors, or a high *node degree*. Although



(a) The Maximum Power Graph.



(b) The reduced topology after the topology construction mechanism.

Fig. 6.3 Comparison of topologies: the MaxPowerGraph versus the reduced topology.

this may be seen as an advantage, it creates two important problems. Firstly, packet transmissions have a higher probability of collision, and secondly, long packet transmissions limit the possibility of frequency reuse.

Figure 6.3 shows an example of a network in which nodes transmit at their maximum power compared with the same network after the topology construction mechanism (Minimal Spanning Tree).

6.2.3 Increased Network Capacity

Topology construction can also have the effect of increasing the network capacity. Reducing the transmission power can eliminate the exposed terminal problem (see Section 3.2). For example, in Figure 6.4 on the left, node B is transmitting to node A and node C has a packet for node D, but C can not transmit the data. If both B and C reduced their transmission power in the appropriate amount, they could be transmitting at the same time (Figure 6.4 on the right) without causing interference to each other.

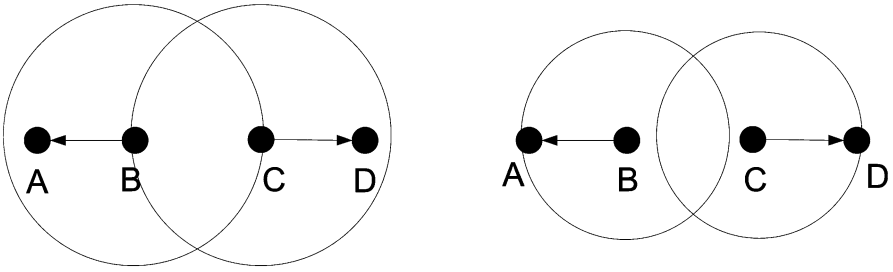


Fig. 6.4 The exposed terminal problem.

6.3 Challenges in Topology Control

Although the energy saving benefits of using topology control are widely accepted, topology control is a complex process that, if not performed carefully, may end up producing undesired results. If the algorithms and protocols designed to perform the topology construction and maintenance functions are not energy efficient, they may end up expending equal or more energy than the energy saved as a result of the reduced topology. Therefore, topology control must be energy efficient as well, a factor sometimes not well evaluated in many research studies.

Topology construction and maintenance mechanisms need to (1) exchange information to make topology control decisions, and (2) run algorithms to make those decisions. The first factor is about communication overhead, which is the most energy consuming factor in wireless sensor networks. Therefore, topology control mechanisms must minimize the use of control packets. The second factor is about computations, or the complexity of the algorithms that need to be run in order to make the topology control decisions. This is a less critical factor as compared with the communication overhead, but important as well. The algorithms need to be simple and scalable so they can be run in wireless sensor nodes.

The literature does not dwell on these two factors, mainly because authors don't look at the entire topology control iterative cycle. Most studies about topology control only consider what we call the topology construction phase. In these studies, authors show that their mechanisms in fact reduce the energy consumption and extend the lifetime of the network, but they usually do not close the cycle as many times as possible. Therefore, the overhead and computations of the topology maintenance phase are not included in most evaluations.

6.4 Design Guidelines

The following are important considerations in the design of effective topology control mechanisms:

- **Distributed algorithm:** Centralized topology control mechanisms need global information and therefore are very expensive to be implemented in practice.
- **Local information:** Nodes should be able to make topology control decisions locally. This reduces the energy costs and makes the mechanism scalable.
- **Need of location information:** The need of extra hardware or support mechanisms adds to the cost in terms of dollars and energy consumption. One example is the need of location information, which might be provided by GPS devices or localization protocols.
- **Connectivity:** The reduced network must be connected, so all active nodes can exchange information among themselves as well as with the sink node.
- **Coverage:** The reduce topology must cover the area of interest.
- **Small node degree:** A small node degree means a small number of neighbors, which at the same time traduces into a lower number of collisions and retransmissions, saving energy. A small node degree may also help to mitigate the hidden and exposed terminal problems.

- **Link bidirectionality:** Bi-directional links facilitate the proper operation of some MAC layer protocols, such as the one utilized by the IEEE 802.11 standard, which sends RTS/CTS signals and also acknowledgments in the return path.
- **Simplicity:** Topology control algorithms must have a low computational complexity, so they can be run in wireless sensor devices.
- **Low message complexity:** Topology control mechanisms must work with very low message overhead, so they are energy-efficient and can be run many times as part of the topology control cycle.
- **Energy-efficiency:** All the factors considered thus far and those discussed in the last section converge in the issue of energy-efficiency, which is essential for topology control mechanisms and wireless sensor networks in general.
- **Spanner:** We want the reduced topology to be a spanner of the Unit Disk Graph in terms of both length and number of hops. A subgraph G' is a spanner of a graph G for length (number of hops) if there is a positive real constant x such that for any two nodes, the length (number of hops) of the shortest path in G' is at most x times of the length (number of hops) of the shortest path in G. The constant x is called the *length (number of hops) stretch factor*.
- **Convergence time:** We want the topology construction process to take place as fast as possible and converge after a limited number of steps.

6.5 Definition of Topology Control

In order to understand how topology control can reduce energy consumption, it is important to clarify first the concept of topology of the network. The *topology of the network* is determined by those nodes and links that allow direct communication. Once a wireless sensor network is deployed, each of the nodes communicates with a subset of the nodes according to the distance between them; communication links are established with those nodes that are close enough for the radio signal to arrive with enough strength to be detected. These nodes and links establish the topology of the network.

A more formal way of representing a wireless sensor network is by using a *Geometric Random Graph*, $G = (V, E, r)$, in which V is the set of vertices, E is the set of edges, and r is the radius of the transmission range of the nodes. Every vertex on V represents a wireless sensor device, and has a geometric coordinate associated to it and an open ball with radius r. This open ball is a set that contains all the vertices with distance less than r respective to the node, which represents the communication area of the nodes. The nodes in the open ball are the only neighbors that the node can communicate with directly. The formal definition of the open ball is presented in Equation 6.4 [94] as follows:

$$B_r(x) = \{y : |x - y| < r\}, \quad x, y \in V \tag{6.4}$$

The set of edges E are the union of all pairs created by each vertex and all the adjacent vertices contained in its open ball. Providing that communication is not always bidirectional, links are modeled as unidirectional edges. Given that the open

balls of the vertices are independent, the existence of the edge (x, y) does not imply the existence of the edge (y, x). In addition, if any metric can be calculated between a pair of adjacent nodes, i.e., distance, angle, remaining energy, etc., it can be used as the weight of the edge. This metric, as well as the edges, is not always symmetric, so weights could be different from one direction to the other. The formal definition of the set of edges is presented in Equation 6.5.

$$E = \{(x, y, d) : y \in B_r(x) \wedge d = |x - y|\}, \quad x, y \in V \tag{6.5}$$

Normally, after deployment, sensors are set to transmit at maximum power in order to communicate with the greatest number of nodes. The resultant graph, called the *Maximum Power Graph*, is taken as the initial graph to work with because it is the maximum topology of the network in terms of active nodes and active links. One example of this graph is shown in Figure 6.3(a), in which 500 nodes are uniformly deployed in an area of 500 m $\times$ 500 m with transmission ranges of 80 m.

Usually, due to the high density and random deployment of the nodes, many sensors are localized very close to each other. In these cases, the sensing information that close sensors provide will tend to be very similar, and therefore, the presence of "redundant" nodes in the network is not entirely necessary. Further, having redundant nodes active in the network is not even desired; they waste additional energy retransmitting packets that collide because of the increased network traffic. One solution to these problems is to modify the topology of the network taking advantage of the sensors transmission power adaptability and their different modes of operation. The proper combination of the transmission power and the status of the sensors allows the network operator to design a low energy consuming topology while preserving the desired sensing and communication connectivity and coverage.

As it was said before, a dense and randomly deployed wireless sensor network may contain many redundant nodes that will consume additional energy unnecessarily. However, if the network operator had control over certain node parameters and could change the mode of operation of the nodes at will, the topology of the network could be changed and the lifetime of the network could be extended considerably. Therefore,

> **Topology Control** is the reorganization and management of node parameters and modes of operation from time to time to modify the topology of the network with the goal of extending its lifetime while preserving important characteristics, such as network and sensing connectivity and coverage.

It is worth emphasizing that the definition of topology control just given considers not only the case of changing the topology by varying the transmission power of the nodes but also turning nodes on and off as needed. Further, the definition goes beyond topology construction by considering topology maintenance, since the status of the topology changes with time and it will need to be updated from time to time.

6.6 Topology Control and the Communications Protocol Stack

A final aspect worth considering before explaining the different topology control mechanisms is their location in the communications protocol stack. The following two questions are important: Where in the protocol stack should the topology control mechanism be located? How does the topology control mechanism interact with the other layers? We believe that the topology control function must be a thin software layer located between the data link layer and the network layer, interacting with both of them, in the minimum case.

After the network initialization phase, the nodes have been deployed and the data link layer has triggered all necessary procedures for the nodes to find their neighbors and establish the corresponding links using the maximum transmission power. Therefore, the MaxPowerGraph is setup. At this point, the data link layer needs to signal the topology control thin layer to start the topology construction phase. During this phase, the topology control thin layer needs to interact with the data link layer in order to change the transmission power of the nodes or put some nodes to sleep according to the specifics of the topology construction mechanism. Once the reduced topology has been built, the topology control thin layer needs to signal the network layer, so it can start the route discovery process and learn how to route packets. Then, the topology maintenance phase takes place to monitor the status of the reduced topology and determine when to change it, whether locally or globally. Whenever the reduced topology needs to be changed, the topology control thin layer needs to signal both the data link layer and the network layer again, in that order. Once the data link layer has changed the topology, the network layer needs to be informed to trigger the route discovery process again. And the process continues following the iterative cycle describe at the beginning of this chapter. The topology control thin layer could also receive a signal from the network layer whenever the routing process finds that an existing route is no longer available. This signal could be used by the topology control thin layer to trigger a new topology construction phase, so connectivity is guaranteed again.

There are other instances where interactions between these layers will be necessary. For example, the Data Link Layer plays an important role discovering new nodes. This may be the case in two situations: (1) in applications with mobile nodes, and (2) in re-deployments, because new nodes are re-deployed to bring the network back to its original operational level. In any case, the topology control thin layer needs to know about the existence of new neighbors before a new topology construction phase is triggered. Here there are two choices: either the DLL signals the topology control thin layer every time a new neighbor is discovered, or the topology control thin layer queries the DLL for a neighbor update before executing the topology construction phase.

6.7 Topology Control Taxonomy and Road Map

Figure 6.5 depicts the classification of topology control mechanisms and provides the road map for the rest of the book. Part II is devoted to topology construction.

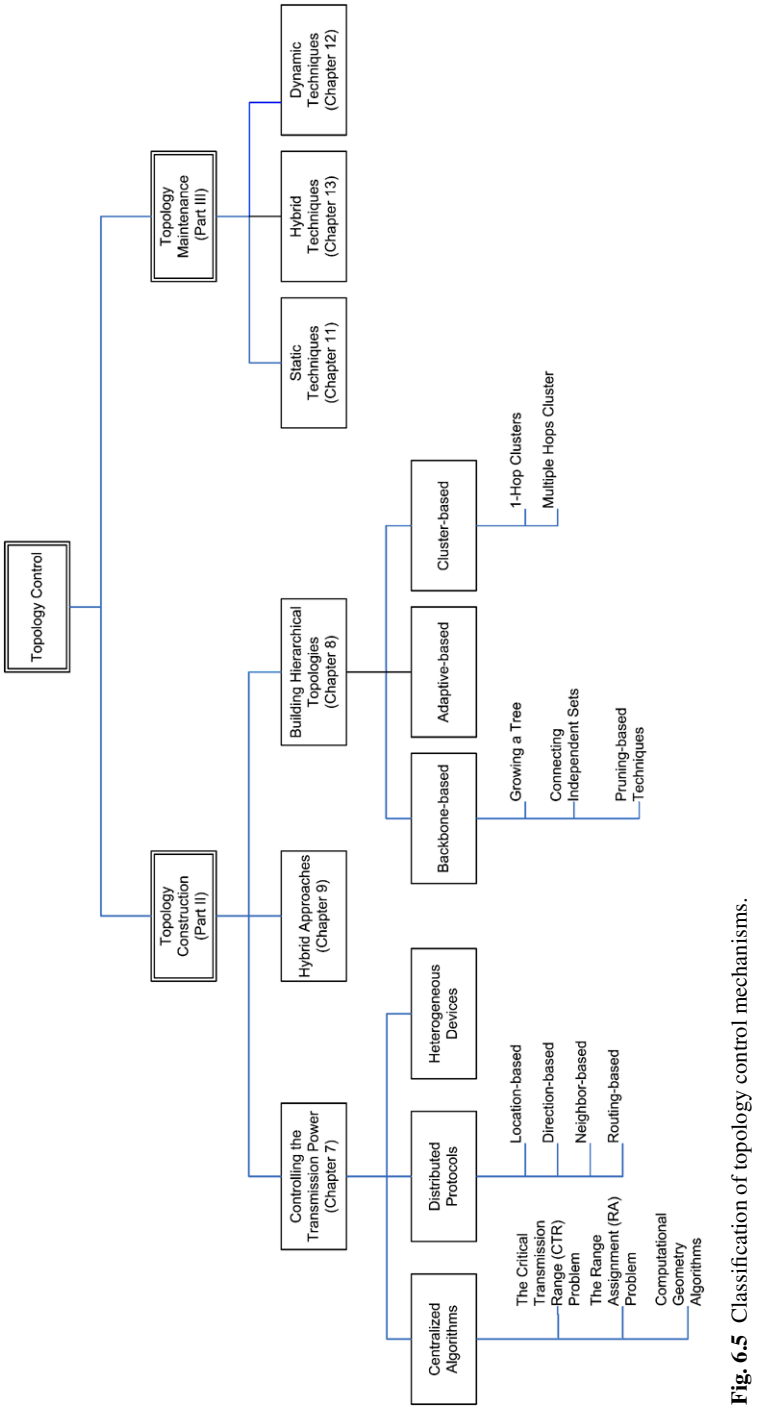


Fig. 6.5 Classification of topology control mechanisms.

Chapter 7 includes techniques used to build a reduced topology controlling the transmission power. Networks of homogeneous and heterogeneous devices are considered. Controlling the transmission power in homogeneous networks can be exercised using centralized algorithms and distributed protocols. Centralized algorithms aim at solving the Critical Transmission Range (CTR) and the Range Assignment (RA) problems. In addition, some useful algorithms from computational geometry are included in the chapter. Chapter 7 also includes important protocols for heterogeneous networks, which are not very common in the literature. Chapter 8 considers the topic of topology construction building hierarchical topologies by means of backbones and clusters. Hybrid approaches are described in Chapter 9.

Part III is about topology maintenance and their performance evaluation. Chapter 11 describes static techniques while Chapter 12 covers dynamic techniques. Static topology maintenance techniques calculate the new and subsequent topologies during the first topology construction phase. The algorithms then switch precalculated topologies according to some criteria. Dynamic techniques find a new topology on the fly, such as running the topology construction mechanism every time a specific criterion is met. Both techniques can be global or local in scope, switching the entire tree, an entire branch or cluster, or a single or few links. Finally, Chapter 13 considers hybrid techniques, which are a combination of static and dynamic techniques, and concludes about the performance of all these techniques.

Part II
Topology Construction

Chapter 7
Controlling the Transmission Power

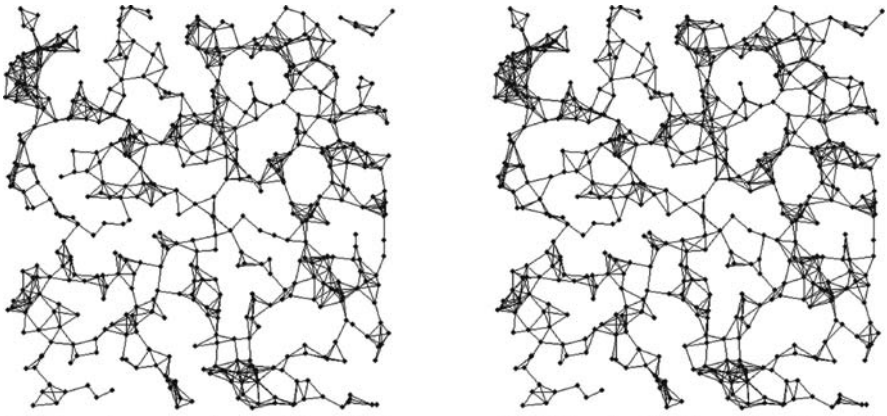
7.1 Introduction

This chapter describes the most important topology construction algorithms and protocols that build the reduced topology by controlling the transmission power of the nodes. First, we consider the case where all nodes in the network are similar, i.e., assuming an homogeneous infrastructure. Here, we have centralized algorithms and distributed protocols. Controlling the transmission power in a centralized manner is perhaps the most mature topology construction technique. Under this approach there are two well-known problems: The **Critical Transmission Range (CTR) problem**, which finds the minimal communication range for all the nodes in the network while preserving network connectivity, and the **Range Assignment (RA) problem**, which goes further in the goal of saving energy finding the optimal transmission power for each individual node. These two problems emphasize the construction of an optimal topology. Distributed topology construction protocols are described next. Here, we are more concerned about building a "good quality" topology while being able to implement it in an efficient fashion, where efficient means with low energy consumption, low computational and message complexity, and so on. These protocols make use of location, direction, neighbor, and routing information to achieve their objectives. Finally, the chapter also includes mechanisms that find the reduced topology by controlling the transmission power in heterogeneous networks, i.e., without assuming similar nodes, which is, in fact, a possible scenario.

7.2 Centralized Topology Construction: The Critical Transmission Range (CTR) Problem

The main goal of this technique it to find the minimum homogeneous transmission range that, applied to every node, will produce a connected graph. This transmission range is called the Critical Transmission Range, or CTR.

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,



(a) The naive approach to find the critical transmission range.

(b) Finding the EMST to reduce the transmission range.

Fig. 7.1 The EMST approach to finding the critical transmission range.

A naive approach to finding the CTR could be to find the largest distance among the closest neighbors of all nodes. However, this distance does not guarantee that the resulting network will be a connected network. One counter example would be two subgraphs connected by one edge e whose distance is greater than the selected range because no node has its closest neighbor at a distance larger than the one in e. If e disappears then the final topology will be a non-connected graph. (See the upper right part of Figure 7.1(a).)

This counter example suggests that an structure that will assure connectivity has to be found first in order to consider the important edges in the decision of finding the largest edge. One approach in this direction is to build an Euclidean Minimal Spanning Tree, EMST, that will provide the minimal cost coverage of all nodes in the graph. If the largest edge from the set of the tree edges is selected, it will assure that no ignored edge will jeopardize connectivity. The result of using this technique is shown in Figure 7.1(b), which shows that the network is now connected.

The problem with techniques based on the EMST approach is that calculating an EMST requires centralized decisions based on exact information about node positions. Even though there are techniques to calculate approximations of EMST in several fashions, like randomized algorithms [60] and local information [73], they usually require the exchange of many messages, which is not desirable in resource constraint platforms like wireless sensor networks. Often these approaches do not produce an optimal energy spanner.

Another approach is to use the theory of *Geometric Random Graphs*, which provides an analytical solution to the communication range problem that, with high probability (*w.h.p.*), produces a connected topology under some assumptions. Let us assume a square area of side l, in which n nodes are deployed uniformly distributed at random. In [90], Penrose determines *w.h.p.* that, for dense networks, the length of the longest edge, from which the value of the CTR can be calculated, is given by

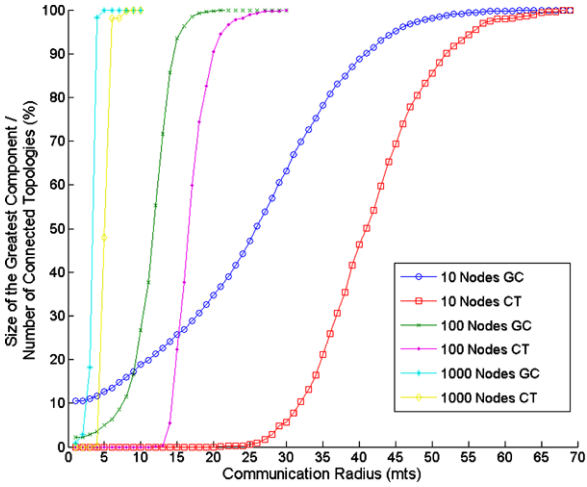


Fig. 7.2 Practical consideration of the CTR: The giant component.

Equation 7.1.

$$CTR_{dense} = \sqrt{\frac{log\, n + f(n)}{n\pi}} \tag{7.1}$$

where $f(n)$ is an increasing function of n, such that $lim_{n\to\infty} f(n) = +\infty$, and $log\, n$ is the natural logarithm of n ($ln\, n$).[1]

Equation 7.1 has several limitations. First, it only applies to dense networks. The asymptotic value of the longest edge is found in a fixed area as the number of nodes goes to infinity. Second, it is not very accurate. Experiments in [105] show that even for a large number of nodes ($n = 2500$), the theoretical value given by Equation 7.1 has a relative difference in the order of 28% when compared with experimental results. Finally, it may not be a good CTR to use in the field, as Equation 7.1 does not say anything about the distribution of the number of nodes connected. This last disadvantage is related to the giant component described in [105, 49], which is further analyzed here next.

Figure 7.2 shows simulation results that further explain the practical implication of using the CTR provided by Equation 7.1. The graph shows the results of three experiments where 10, 100, and 1000 nodes are uniformly distributed in an area of 100 m × 100 m, and function $f(n) = log\, log\, n$. Two curves are included per experiment, the giant component (GC) curve, and the connected topologies (CT) curve. Both curves are drawn as a function of the communication radius, which is increased in steps of 2 meters per experiment. Each experiment point is the average of running 1000 topologies. The GC curve provides the percentage of the total number of nodes that are contained in the largest connected set. So, for example, if we look at the curves in the middle (experiment with 100 nodes), the curve on the left says that with a communication radius of 11–12 m, the greatest connected set contained

[1] Unless otherwise specified, all logarithms in this section are natural logarithms.

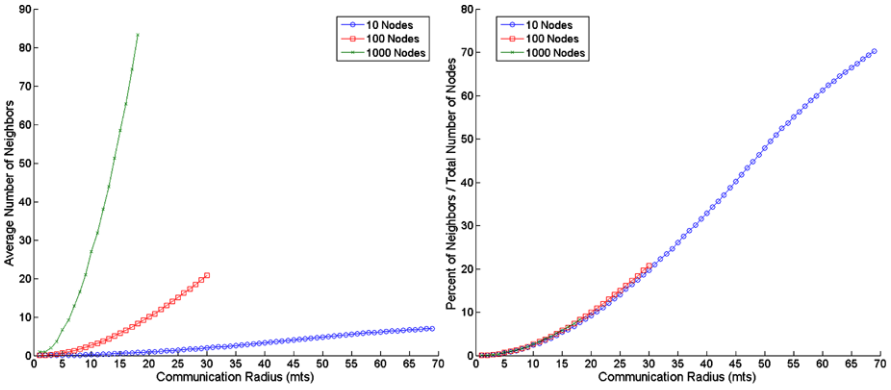
Table 7.1 Giant component and connected topologies results using simulations versus number of CTRs given by Equation 7.1.

Number of CTRs	Number of Nodes	Theoretical CTR (Equation 7.1)	Giant Component (Simulations)	Connected Topologies Simulations
	10	31.59	69.87	10.30
1 × CTR	100	13.97	85.69	5.40
	1000	5.30	99.99	98.10
	10	47.38	97.36	80.50
1.5 × CTR	100	20.95	99.88	94.50
	1000	7.95	99.90	99.80
	10	63.18	99.90	99.00
2 × CTR	100	27.94	99.90	99.80
	1000	10.6	100.00	100.00

57–58% of the nodes. The CT curve, on the other hand, shows the percentage of the 1000 topologies that were 100% connected. So, for example, the middle curve on the right, shows that if a radius of 17 m is used, 60% of the topologies were 100% connected.

The most important conclusion that can be derived from these curves is that a very large connected component is built very quickly using a communications radius considerably smaller than the radius needed to have the entire network connected. For example, continuing with the curves in the middle, a radius of 15 m builds a connected set with nearly 97% of the nodes while a radius of 22 m is needed to have 97% of the topologies connected, and a radius of 27 m (almost twice as large) is needed to have 100% of the topologies connected. The practical implication of this observation is that sometimes a completely connected topology is not needed, especially in dense networks, and therefore we can use a much smaller transmission power. Put in a different perspective, if a fully connected network is needed with high probability, the transmission power must be set to a very large value; however, if a few disconnected nodes can be tolerated, this power can be substantially reduced. The second (expected) observation is that the communication radius decreases with the density of the network.

Using the same simulation experiment results, Table 7.1 shows the accuracy of Equation 7.1 versus experimental data for different network sizes (in number of nodes), and the effect of increasing the CTR. Several interesting observations can be made. First, as in [105], it can be seen that the CTR given by Equation 7.1 is not very accurate, especially in small networks. Remember that the CTR is the minimum value that should produce a connected topology; however, the experimental results show that only 10.3% of the 1000 topologies were actually connected. Second, it is also observed that the accuracy increases with the number of nodes. Finally, it is interesting to observe similar results increasing the CTR by 1.5 and 2 times. As expected, the accuracy now increases with the CTR. In the case of small networks (10 nodes), 80.5% and 99.0% of the 1000 topologies were connected with a CTR of 1.5 and 2 times the CTR, respectively.



(a) The absolute value of the node degree.

(b) Node degree divided by the total number of nodes.

Fig. 7.3 Analysis of the node degree.

Another interesting aspect of these experiments is the node degree of the network. Figure 7.3(a) shows how the absolute value of the node degree increases with the communication radius regardless of the network density, which is the expected behavior. Similarly interesting, and with practical implications, is the result shown in Figure 7.3(b), which says that, in a wireless sensor network with uniformly distributed sensors, for a given communication radius, the node degree divided by the total number of nodes is the same regardless of the network density.

The results presented thus far have been derived for 2-dimensional networks using a uniform distribution for the node deployment and a large number of nodes (dense network). Similar equations have been derived for variations of these assumptions. For example, using theorems from [50, 90, 92], in [105] the CTR for the one-dimensional case using the uniform distribution for the node deployment is shown to be:

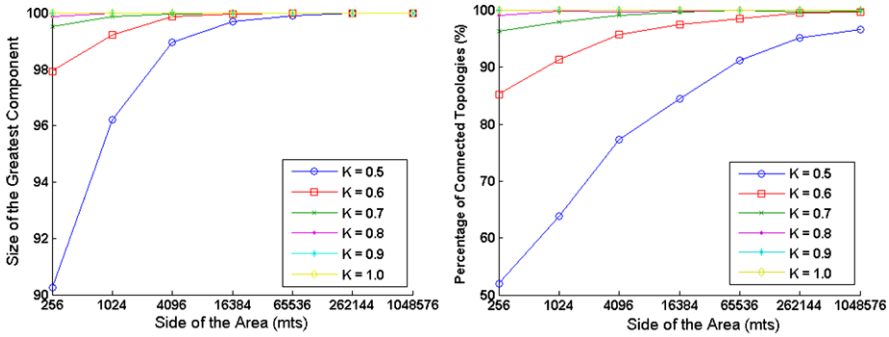
$$CTR_{dense} = \frac{log\, n}{n} \tag{7.2}$$

Similarly, using theorems from [31, 93], [105] presents the CTR for the 3-dimensional case in a uniformly distributed deployment as:

$$CTR_{dense} = \sqrt[3]{\frac{log\, n - log\, log\, n}{n\pi} + \frac{3}{2} \times \frac{1.41 + g(n)}{n\pi}} \tag{7.3}$$

where $g(n)$ is an arbitrary function such that $lim_{n\to\infty} g(n) = +\infty$.

The CTR for sparse networks has also been investigated. Here, the asymptotic behavior is found relaxing the assumption of a fixed deployment area, and the CTR is found for those values of the transmission range r and number of nodes n that make the network connected *w.h.p.* when $l \to \infty$. Actually, these theoretical results can be applied to both dense and sparse networks, depending on the relative magnitude of n and l. Different theorems in [106] show the CTR for connectivity in sparse



(a) The giant component. (b) The number of connected topologies.

Fig. 7.4 The CTR for sparse networks.

networks under the assumption of a uniformly distributed random node deployment for one-, two-, and d-dimensional networks. For the one-dimensional case, the CTR is given by:

$$CTR = k\frac{l\,log\,l}{n} \tag{7.4}$$

where k is a constant with $1 \leq k \leq 2$, and l is the length of the line. The asymptotic behavior of this CTR has been shown to be very accurate when compared with experimental results in [105] for $k = 1$, if certain assumptions are considered. For instance, the relative magnitude of the transmission range r and the number of nodes n when expressed as a function of the length of the line l have to be such that $r = r(l) \ll l$ and $n = n(l) \gg 1$.

For d-dimensional cases, with $d = 2, 3, \ldots$, Santi [105] proposes a partially proved result to find the CTR for connectivity as:

$$CTR = k\frac{l^d\,log\,l}{n} \tag{7.5}$$

where k is a constant with $0 \leq k \leq 2^d d^{\frac{d}{2+1}}$.

The accuracy of Equation 7.5 is validated by the results obtained after several simulation experiments using the same scenarios described in [106]. Assuming a communication range $r = k \cdot l^{3/4} \cdot \sqrt{log_2 l}$, with $0.5 < k < 1.0$, $n = \sqrt{l}$ and $l = 2^{2i}$, where $4 < i < 8$, 1000 random topologies were generated for different parameters of l and k, and results were collected to determine the giant component and the average ratio of connected topologies, and the average node degree of the vertices in the graph, which are shown in Figures 7.4 and 7.5, respectively.

The results in Figure 7.4(a) show how Equation 7.5 with $k > 0.7$ produces a giant component that includes over 98% of the nodes with over 90% of the topologies connected (Figure 7.4(b)) while a value of k equal or close to 1 produces 100% values. However, for values of $k < 0.7$, the figures show that the CTR given by Equation 7.5 does not produce as accurate results. These results should not be seen in

Table 7.2 Values of l, n, and r to produce 100% connected topologies ($k = 1$).

Side of Area (l)	Number of Nodes (n)	Transmission Range (r)
256	16	181
1024	32	572
4096	64	1773
16384	128	5418
65536	256	16384
262144	512	49152
1048576	1024	146543

an isolated manner though, as practical considerations may also impose limitations. For example, Table 7.2 includes the values of l and n utilized in the experiments to produce sparse networks for $k = 1$ while conforming to the stated assumptions, and the value of r, the transmission range of the nodes. As it can be seen, in order to produce 100% connected topologies the transmission range of the nodes must be increased accordingly, and in most of the cases, need to achieve range transmission values way out of the possibilities found in real wireless sensor devices.

Figure 7.5 shows the results of the average node degree. As it can be appreciated, the curves show an increasing trend with l and n, also displaying different increasing rates according to the value of k; the greater k, the faster the increase. The figure also shows the theoretical value of the node degree calculated using results from [138] where it is proved that in order to have a connected topology *w.h.p.*, the average number of neighbors K must be equal to $c \cdot \log n$, with $c = 5.8$. From the figure, two main comments can be made. First, it can be seen that the theoretical value provides an upper bound, although a very coarse one for small values of k. Second, the average node degree is a fairly high value given the number of nodes in the network in each case. Table 7.3 provides more detailed results, where

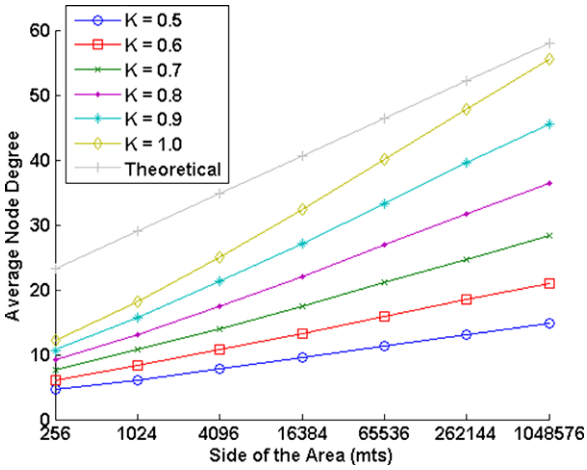


Fig. 7.5 The average node degree for sparse networks.

Table 7.3 Average node degree of the network for different values of l, n, and k.

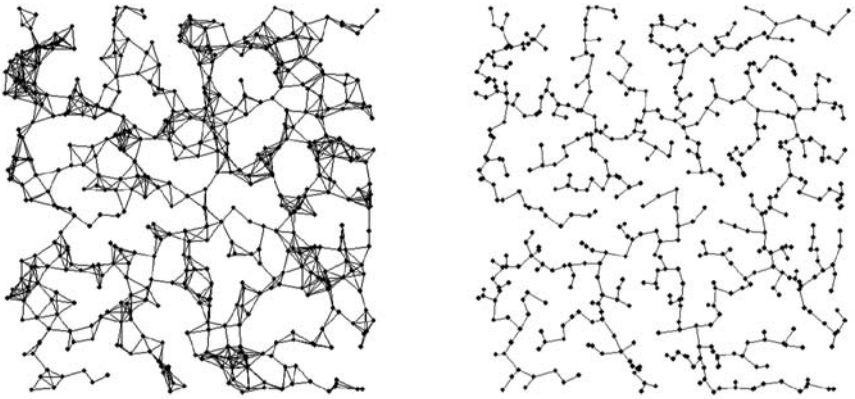
Side of Area (l)	**Number of Nodes** (n)	**Average Node Degree**					
		$k = 0.5$	$k = 0.6$	$k = 0.7$	$k = 0.8$	$k = 0.9$	$k = 1.0$
256	16	4.62	6.10	7.65	9.24	10.69	12.11
1024	32	6.14	8.30	10.76	13.06	15.66	18.12
4096	64	7.76	10.71	14.00	17.50	21.26	24.95
16384	128	9.51	13.28	17.53	22.06	27.12	32.32
65536	256	11.25	15.88	21.09	26.92	33.22	40.07
262144	512	13.04	18.45	24.72	31.74	39.48	47.85
1048576	1024	14.80	21.03	28.28	36.47	45.60	55.55

it can be seen, for example, that in the case of $l = 256$ and $n = 16$, the average node degree goes from 4.6 for $k = 0.5$ to 12.11 for $k = 1$. This, again, may have practical implications. If a 100% connected network is desired, according to the results given in Figure 7.4, a value of k equal or close to 1 must be chosen; however, with such value, the average node degree is fairly high, an undesirable case given that it defeats several of the goals of topology control. For example, a network with a high node degree is very well connected, maybe close to the connectivity of the MaxPowerGraph, which is the graph topology control is meant to optimize. Second, many neighbors imply more collisions, and therefore, more energy wasted, and less possibility for frequency reuse.

Finally, the CTR for connectivity in irregular areas with different node deployment distributions has also been considered although it is still an active area of research. For example, the use of the normal distribution is of particular interest given that it reflects the practical deployment of sensors in many applications, such as deployments from an airplane. In [91] the CTR for connectivity in two- and three-dimensional networks with normally distributed nodes has been derived.

7.3 Centralized Topology Construction: The Range Assignment (RA) Problem

The procedure to find the Critical Transmission Range may be a hard and costly operation. Also, in some cases, the CTR might be very close to the maximum transmission range, and therefore, there will be neither major differences in the topology nor energy savings. Further, sometimes, the solution utilized to find the CTR provides an even better solution if the homogeneous constraint is removed. For example, if the EMST procedure is used, it can be seen that each node in the resulting graph is connected to its neighbors with different ranges, and some of them are really small, so there is no need for them to use a higher transmission power in order to be connected. Figure 7.6 shows the differences in the reduced topologies when using the EMST to solve the CTR and the RA problems. However, in reality this is not as good as it looks. It was said before that finding the EMST is a very expensive task in terms of overhead and energy. Also, a small neighborhood might not guar-



(a) Reduced topology using the EMST to solve the CTR problem.

(b) Reduced topology using the EMST to solve the RA problem.

Fig. 7.6 Reduced topologies using the EMST approach to solve the CTR and RA problems.

antee connectivity if there are unexpected changes in the network, so having more neighbors is not totally undesirable in benefit of the reliability.

A more interesting and energy-efficient method would be to find the maximum energy needed per node, a *non-homogeneous power assignment* to build a reduced connected topology without these drawbacks. This is the well-known Range Assignment (RA) problem, which is defined as:

> *The Range Assignment (RA) problem for a set of n nodes in the d-dimensional space is the function that finds a strongly connected graph and minimizes the total cost of the network, which is given by the summation of the transmission power used by all n nodes.*

As it can be seen, the CTR problem is a special case of the RA problem in which the nodes are constraint to use the same transmission power. However, relaxing this simple constraint makes the RA problem more complex. First, the computational complexity of the algorithms that solve the RA problem is considerably higher. For example, while the CTR problem in one-dimensional networks can be solved in $O(n \log n)$ time, the algorithm presented in [64] to solve the RA problem in similar networks has a computational complexity of $O(n^4)$. Solving the RA problem in two- and three-dimensional networks is NP-hard [64, 24]. Even easily computed approximations, such as building a Minimum Spanning Tree and finding the range assignments, has an $O(n^2)$ time complexity. Second, solving the RA problem, as defined before, may produce unidirectional links in the topology, as nodes have different transmission ranges. This is important because we are interested in having a strongly connected topology and these unidirectional links might be essential for ensuring such connectivity. Therefore, symmetry constraints have been added to the RA problem, and two new problems emerged: The **Weakly Symmetric Range Assignment (WSRA) problem** and the **Symmetric Range Assignment (SRA) problem**.

The solution to the Weakly Symmetric Range Assignment (WSRA) problem removes unidirectional links and determines the range assignment for each node such that the reduced graph is connected and symmetric, and the total cost of all the assignments is minimum. Note that in the case of the WSRA problem, the resulting topology may still contain some unidirectional links, which are not needed for connectivity. The Symmetric Range Assignment (SRA) problem, on the other hand, is stronger in its requirements, as all links in the resulting topology must be bidirectional. This implies that some nodes will have to increase their transmission power to produce the symmetric topology.

There are several reasons why it is more convenient to solve the WSRA problem in wireless sensor networks. First, the algorithms used to solve the RA, WSRA or SRA problems in two-dimensional networks present the same computational complexity. Second, the WSRA problem creates a connected backbone of bidirectional links, which is what nodes need to communicate. Additional unidirectional links can be ignored without any connectivity penalty. Third, it has been proved in [105] that the energy cost of the WSRA problem has a marginal additional energy cost compared with the RA problem; however, a weakly connected topology guarantees a bidirectional backbone essential for the correct operation of many data link and network layer protocols. In comparison, the stricter restrictions imposed by the SRA problem do have a significant energy cost.

7.4 Algorithms from Computational Geometry

In this section we describe the Relative Neighbor Graph, the Gabriel Graph and the Delaunay Triangulation, results from computational geometry that can be used to find reduced graphs. Where to include these results in the general topology control taxonomy is somehow difficult. It was decided to include them here because they can all be implemented in a centralized manner. However, several distributed implementations exist. They could have been included under the location-based distributed protocols category as well, as they all assume that information about distances between nodes or relative positions is available.

The **Relative Neighbor Graph (RNG)** [121] eliminates the longest edge from every triangle formed by two of its neighbors and itself. Formally, the RNG $G' = (V, E')$ of a graph $G = (V, E)$ is defined as:

$$\forall u, v \in V : (u, v) \in E' \text{ iff } \neg\exists w \in V : max\{d(u, w), d(v, w)\} < d(u, v) \quad (7.6)$$

where $d(u, v)$ is the Euclidean distance between two nodes. The RNG can be easily determined using a local algorithm with message complexity $O(n)$ and computational complexity $O(n^2)$. Also, if the original graph G is connected, then G' is also connected. However, nodes that are a few hops away in G can become very far apart in G'. Relative neighbor graphs have an average node degree of 2.6 [18].

In the Euclidean plane, the **Gabriel Graph (GG)** [38] connects point u and v if the disk having line segment uv as its diameter contains no other node than itself

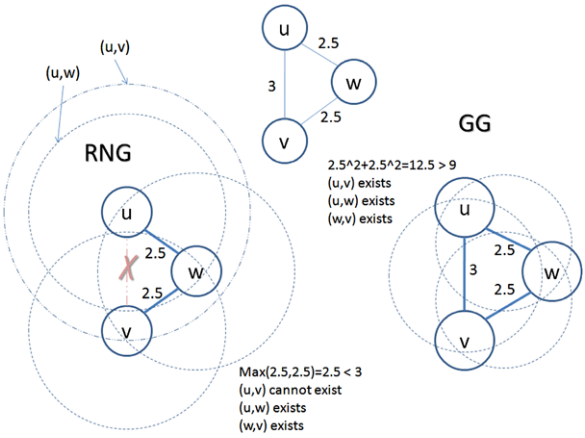


Fig. 7.7 Example of RNG and GG algorithms.

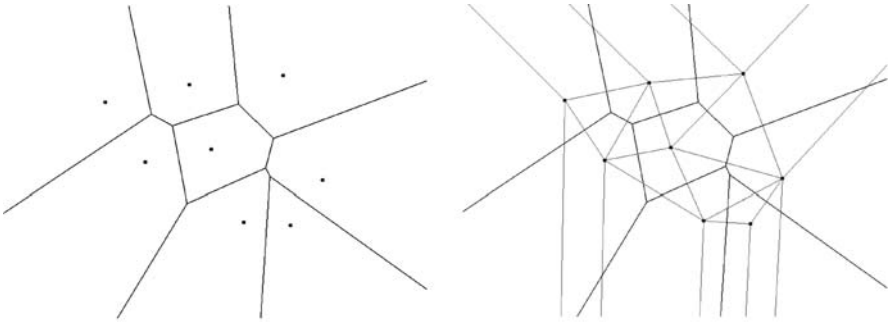
and the neighbor. Formally, the GG $G' = (V, E')$ of a graph $G = (V, E)$ is defined as:

$$\forall u, v \in V : (u, v) \in E' \text{ iff } \neg\exists w \in V : d^2(u, w) + d^2(v, w) < d^2(u, v) \quad (7.7)$$

where $d(u, v)$ is the Euclidean distance between nodes u and v. The GG also maintains connectivity and has the same message and computational complexity of RNG.

As it can be seen RNGs and GGs are very similar; they both remove every link to a neighbor node that could be reached through another neighbor. Distributed implementations of the RNGs and GGs only require nodes to share their locations with their neighbors and test these conditions to verify each edge in order to determine the minimal set of neighbors. For example, a distributed version of the GG can be found in [62]. Although these two graphs have low message complexity ($O(n)$), their node degree can be as high as $n - 1$. If the node degree is not upper bounded, bottlenecks may exist in the communication graph. Figure 7.7 shows how these techniques, although very similar, produce different final topologies. In the figure, the weights on the edges are given by the Euclidean distance between the respective nodes.

Another graph borrowed from computational geometry that is useful to build reduced topologies is the **Delaunay Triangulation (DT)**. If we connect all the neighbor nodes based on the vicinity of the Voronoi diagram we will have a Delaunay Triangulation. The Voronoi diagram is a geometric construction that defines the area of coverage and the vicinity in a graph. Trace an imaginary line between two nodes. Right in the middle of this line, trace an orthogonal line that will define the limit between the areas. Repeat the process between every pair of nodes in the graph. The smaller intersected area defined around a node from the limits with every other node in the graph determines the final diagram (see Figure 7.8(a)). The Delaunay Triangulation diagram is formed by connecting adjacent vertices in the Voronoi diagram, as depicted in Figure 7.8(b). With this approach each node can choose as its



(a) The Voronoi diagram. (b) The Delauny Triangulation diagram.

Fig. 7.8 Relationship between the Voronoi diagram and Delaunay Triangulation.

transmission power the power needed to reach its farthest neighbor. A Java-based application to create and draw Voronoi and Delaunay Triangulation diagrams can be found in [22].

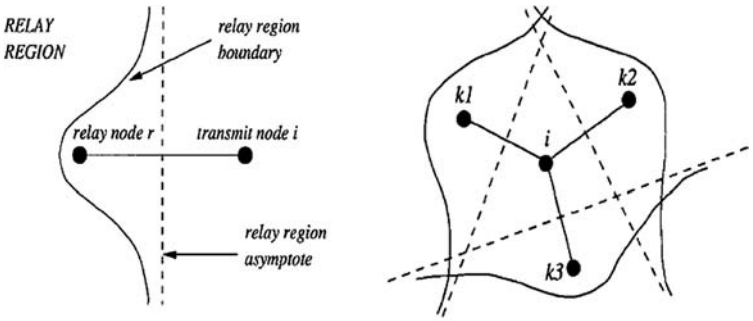
This approach requires global information and, if applied without restrictions, it may connect nodes which are more distant than the maximum communication range. DT has a message complexity of $O(n)$ and computational complexity of $O(n \log n)$. It also guarantees connectivity and may have a node degree as high as $n - 1$. In [75], a localized version of the Delaunay graph is described.

7.5 Distributed Topology Construction for Homogeneous Networks

The last two sections of this chapter described algorithms that find an optimal reduced topology in a centralized manner. The main drawbacks of these algorithms are known to be their centralized approach and complexity. In this section, more practical implementations are discussed presenting topology construction protocols that work in a distributed manner, still in homogeneous networks. As you will see, these protocols are not meant to produce optimal topologies; they rather focus on producing "good quality" topologies while being efficient in terms of energy consumption and computational and message complexity. These protocols, which were classified in Figure 6.5 according to the type of information they utilize as Location-, Direction-, Neighbor- and Routing-based, are described next.

7.5.1 Location-Based Techniques

The algorithms that solve the topology construction problem using location-based techniques assume that every node knows its own position with a high degree of certainty. This information allows the use of geometric properties to determine the best configuration of the topology in terms of distance between nodes, which at the end



(a) The relay region of node i. (b) The enclosure of node i.

Fig. 7.9 The R&M protocol. Reproduced from [102] © 1999 IEEE.

determines the best transmission range for each of them. For example, distributed versions of the algorithms from computational geometry described in Section 7.4 can be included here. In the following, we describe two important location-based protocols, the R&M and the LMST protocols.

The **R&M protocol** [102] is a location-based protocol based on the concepts of *relay region* and the *enclosure* of a node. If node i wishes to transmit information to node j and there is another node r which can serve as a relay node, the aim of R&M is to transmit the information from i to j with minimum total power. In the calculation of total power, R&M includes not only the power spent by the transmitting node but also the power incurred by the receiving node(s). By varying the position of node j, R&M finds under which conditions it consumes less power to transmit the message through r. Therefore, the relay region of node i is defined as the region where it is more energy efficient to use r than transmit the message directly to j. If node i calculates its relay region with all its neighbors, the region given by the union of the complement of the individual relay regions is the enclosure of node i. Figures 7.9(a) and 7.9(b) show the relay region and the enclosure of node i. It is worth mentioning that additional nodes, that i may find in the relay regions of previously found nodes, can be eliminated from i's neighborhood, as it is more power-efficient to transmit to those nodes through the relay nodes previously found than doing it directly. Based on these concepts, the authors build a sparse graph called the enclosure graph, which they show it is strongly connected if every node is connected to the nodes within its enclosure. Further, they also show that the resulting topology is a minimum power topology.

The R&M protocol is executed in two phases. In the first phase, every node finds its enclosure. During this phase, node i determines the neighbors that it can reach using its maximal transmission power and keeps only those that do not lie in the relay regions of previously found nodes. At the end of phase one, the enclosure graph is configured. In the second phase the protocol utilizes the Bellman–Ford shortest path algorithm on the enclosure graph to find minimum energy reverse spanning tree rooted at the sink node, which is found using the power consumption as the

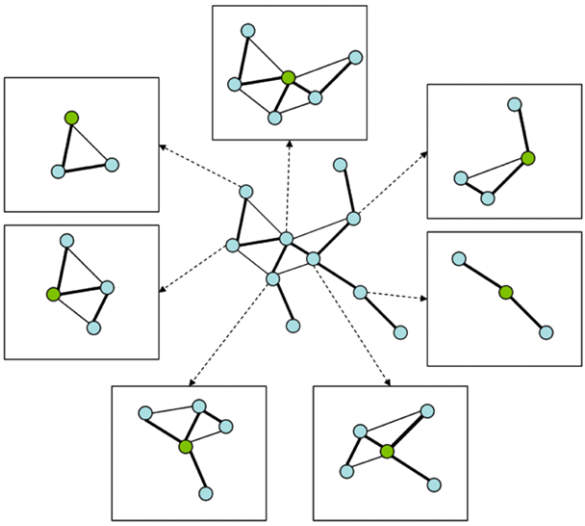


Fig. 7.10 Example of a topology after using the LMST algorithm.

cost metric. Once the minimum power paths are established, data transmission can start. Although the R&M protocol preserves the network connectivity in the worst case, builds a sparse network and an energy-efficient tree, and can be computed in a distributed manner, all nice features for a topology construction mechanism, this second phase may introduce considerably overhead, as the computation of the final topology requires global information.

The **Local Minimum Spanning Tree (LMST) protocol** [73] is another relevant location-based topology control protocol that works in a distributed manner and only needs local information about the topology. In LMST, each node obtains the location from its direct neighbors and creates a local version of the EMST. Then the node selects as gateways the neighbor nodes that are linked with it on the EMST, adapting its range to reach the farthest one. Figure 7.10 shows how each independent node calculates its local MST and how the reduced topology is built. Each node in LMST uses Prim's algorithm [99] to find its LMST along with a weight function that breaks link weight ties to guarantee a unique solution. The weight of the links is given by the Euclidean distance between the nodes, which can be calculated once the nodes have the location of its neighbors.

The authors of LMST showed that links in the topology derived by LMST, G_0, may be unidirectional, and based on that, they proposed two versions of the algorithm. The first version, G_0^+, forces all links to be bidirectional by notifying the neighbors to include the reverse path. The second version, G_0^-, just deletes all unidirectional links. While G_0^- is easier to build and provides a simpler topology, G_0^+ provides additional routes.

In [73], LMST is shown to have the following nice properties:

- Bounded node degree equal or less than 6 not only for G_0 but also for G_0^- and G_0^+.
- Network connectivity is preserved. If the original graph is connected, G_0 is also connected. Further, G_0^- and G_0^+ are also connected.
- Distributed protocol.
- Message complexity of $O(n)$ and computational complexity of $O(m \log n)$, where m is the number of edges.
- Local information.
- Bidirectionality. The two LMST versions produce undirected graphs.

Another location-based algorithm is the **Fault-tolerant Local Spanning Subgraph (FLSS) protocol** presented in [72]. FLSS is a variation of the LMST algorithm described before. FLSS adds edges to a node in increasing order until it creates a k-connected spanning tree in the neighborhood. As such, FLSS preserves k-connectivity. After finding the connected component, an optional procedure adjusts the node's transmission range up to the point in which all edges are bidirectional.

The FLSS algorithm uses its centralized version, Fault-tolerant Global Spanning Subgraph (FGGS), to build the local spanning subgraph. The paper shows that using network flow techniques, a query on whether two vertices are k-connected can be answered in $O(n+m)$ time for any fixed k, where n is the number of vertices and m is the number of edges in the graph. For $k \leq 3$, there also exists $O(1)$ time algorithms. As a result, the time complexity of FGSS is $O(m(n+m))$, and can be improved to $O(m)$ for $k \leq 3$.

Another location-based protocol is the **Geographical Adaptive Fidelity (GAF) algorithm** presented in [137]. GAF is designed based on several observations: First, nodes spend considerable energy sending and receiving packets, and also while in idle state. In order to save more energy, radios must be turned off to conserve energy due to overhearing and idle listening. Second, GAF utilizes network and application layer information to turn the radios off, as these layers provide better information about when the radio is not needed. Third, in highly dense networks, multiple paths exist between nodes, so some intermediate nodes can be turned off without compromising connectivity.

GAF utilizes location information, through a GPS system or any other localization method, to determine node density and redundancy. The deployment area is divided in cells of side r, with $r \leq MaxTxRange/\sqrt{5}$. This cell size guarantees that any node in the cell is able to communicate with nodes on adjacent horizontal and vertical cells. In GAF, each node associates itself with a virtual grid, in which all nodes within the same cell are equivalent in terms of packet forwarding. Equivalent nodes within each cell determine who will sleep and who will forward packets on behalf of its cell neighbors, and for how long. The selection algorithm, which is performed periodically, is based on a ranking procedure based on the state of the node, its expected lifetime (remaining energy level), and its node id (to break ties). This procedure guarantees fairness in terms of traffic forwarding (load balancing) and energy consumption.

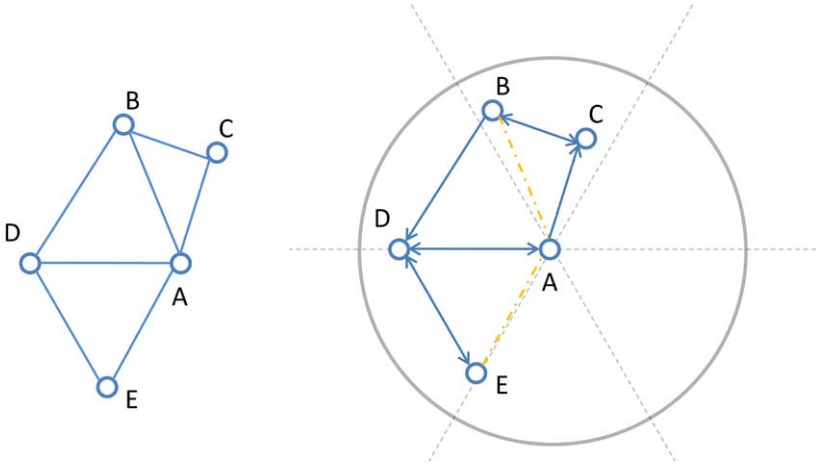


Fig. 7.11 Example of the first phase of the Yao Graph for $k = 6$ cones.

7.5.2 Direction-Based Techniques

In direction-based techniques, it is assumed that the nodes are able to determine the direction of the signals received from their neighbors, and in some cases, the distance between them. The direction of the incoming angle of the signal in the circular communication range can be provided by directional antennae installed in the nodes. Distance information, on the other hand, can be obtained using different techniques like Received Signal Strength Indicator (RSSI), Time of Arrival (TOA), Time Difference of Arrival (TDOA), or any other similar technique. More information about localization techniques can be found in [89, 107, 120].

The algorithm presented in [139], called **Yao Graph (YG)**, is an early version of a direction-based topology control technique in multidimensional spaces. This algorithm works in two phases: (1) reduce the original graph into a subgraph that contains a MST, and then (2) remove the extra edges that will leave only the MST. The subgraph produced after the first stage of the process is called a Yao Graph.

Assume that from each node v in V, there are $k \geq 6$ equally distant rays originating from v forming k cones. The selection of the number of cones determines that each cone has at most an angle $\theta \leq 60°$ or $\theta \leq \pi/3$, which guarantees not only that each node will have at most k neighbors, but that any distance between nodes inside the cone is not the longest edge of the triangle formed by the node in the center and both its neighbor nodes. This characteristic is useful for the construction of the MST. Figure 7.11 shows an example of the Yao Graph of a small MaxPower topology.

After the cones have been established, node v surveys its neighbors on each one of the k cones. Node v calculates the distances from all the neighbors and selects the shortest edge from each cone, creating an undirected edge (v, u), where node u is the closest node to v in that cone. If there is any kind of tie in the closest neighbors selection, any methodology like random selection or using the node ID

can be used to break the tie. The authors prove that a MST is contained in the set of edges resultant from this first stage.

In the second phase, in order to calculate the MST, a global and centralized processor is needed to solve the post-office problem, which determines, for each point $v \in V$, the closest point $u \in V$ such that $d(v, u) < d(v, w), \forall w \in V, w \neq u$. The complete MST algorithm has a complexity of $8 \cdot f(n) + O(n \log \log n)$, where $f(n)$ is the cost of finding the shortest edge on a cone.

Given that in $G = (V, E')$, E' contains a MST, the topology is connected. However, it is important to mention that this algorithm does not guarantee connectivity if the transmission range of the selected neighbor is not greater than the transmission range of the evaluating node. An extension of the Yao Graph that overcomes the heterogeneous communication range problem is presented in [74].

The YG protocol is shown to have the following nice properties:

- Bounded node degree equal or less than k, where $k \geq 6$ cones.
- Network connectivity is preserved, as long as nodes are homogeneous.
- Distributed (Yao Graph) and centralized (MST) protocol.
- Local information, at least during the first phase.

The **Cone-Based Topology Control (CBTC) protocol**, presented in [69, 70], is another implementation of a cone-based topology construction algorithm. Compared with the Yao Graph, CBTC neither calculates the distance between nodes and neighbors nor works with predefined cone sections. In order to avoid the distance calculation, CBTC works dividing the transmission power in a discrete finite set of levels. Then, the node transmission range is changed from level to level. In CBTC, each node u defines an empty circular area of 2π radians and the angle of the cone equal to a parameter α. Each new neighbor that is added to the neighbors list of u will cover a sector of this area, as shown in Figure 7.12. Once the union of all the areas covered by the cones of the neighbors is greater than or equal to 2π, the area of u is said to be covered. The selection of the parameter α becomes very important to guarantee connectivity in the final reduced topology. As it can be seen, this algorithm is more general than the Yao Graph, and probably avoids cases in which YG will not find neighbors in a given sector that might be covered by other nodes in adjacent sectors.

The CBTC protocol also works in two phases. First, it finds a small neighborhood whose union of cone-like sections of each node covers all 2π radians of the circumference from the central node. Second, it removes redundant links without affecting connectivity. The first phase starts with each node u sending a beacon message using the minimum transmission power and receiving replies from all the neighbors that were reached. The nodes that replied are added to the neighbors list of u. If the union of all the areas covered by the cones of the neighbors is less than 2π, then the area of u is said to be incompletely covered, so node u must increase its transmission range and send another beacon message. This process is repeated until the node uses the maximum transmission range or until its area is completely covered.

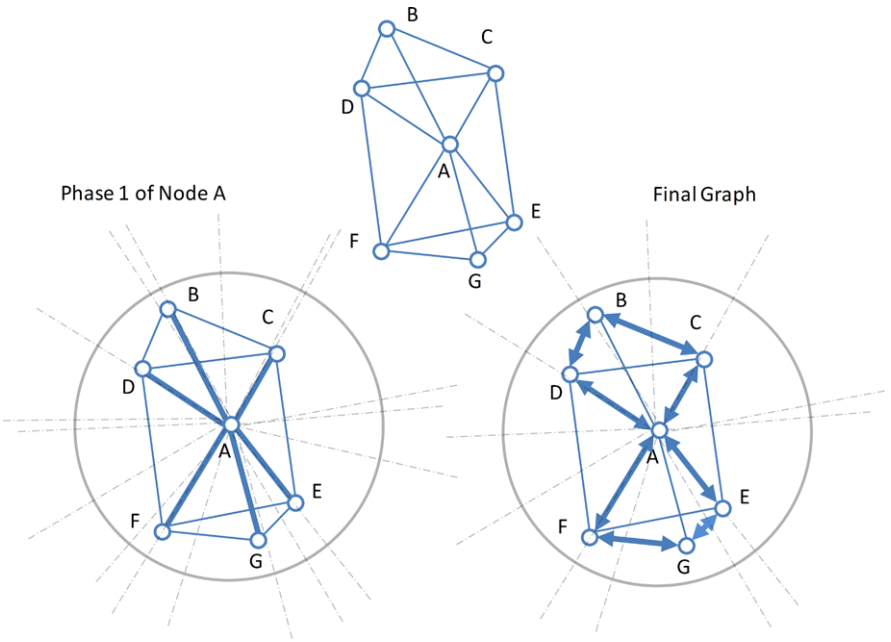


Fig. 7.12 The CBTC protocol.

For the second phase, the authors propose a selection policy defined as follows: Let nodes $v, w \in N(u) \wedge w \in N(v)$, and p be the transmission power, if $p(u, v) + p(v, w) < p(u, w)$, then node u will eliminate node w from $N(u)$.

In [70], a more detailed analysis of the algorithm and some optimization options are presented. One of the optimizations deals with transmission range reduction when a node has a sector with no neighbors. Each node registers the power of the received replies from the neighbors on each sector. Once the maximum power is reached and a gap is detected, the node decreases its transmission power eliminating the now unreachable neighbors, while coverage on the sectors remains the same, i.e., it keeps neighbors on the sectors that were not empty before. One characteristic of this algorithm is that it might have asymmetric links, which is not desirable in most cases. The authors propose two methodologies to deal with this problem: AugmCBTC and RemCBTC.

The first option, AugmCBTC, adds links to the reduced graph from the second phase. In this case each node advertises its own transmission power in order to force all its asymmetric neighbors to increasing their transmission range, until it equals the range of the requester. This graph will increase the average number of neighbors, but will guarantee bidirectionality. The counterpart of the first option, RemCBTC, removes all asymmetric links, reducing the actual number of neighbors of a node, and probably the final transmission power. The authors proved that with $\alpha < 5\pi/6$ and $\alpha < 2\pi/3$ the topology remains connected after the execution of AugmCBTC and RemCBTC, respectively.

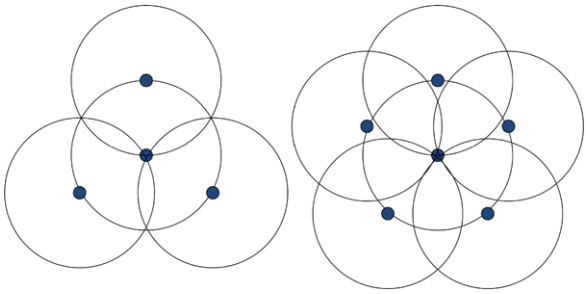


Fig. 7.13 Area of coverage on ideal location with 3 and 5 neighbors.

The CBTC protocol has the following properties:

- Bounded node degree equal or less than 6.
- Network connectivity is preserved in both AugmCBTC and RemCBTC.
- Distributed protocol.
- Local information.
- All links generated are bidirectional.

Another implementation of a direction-based protocol is the **Distributed Relative Neighbor Graph Protocol (DistRNG)** [12], which is a distributed implementation of the original RNG algorithm. As in the original RNG algorithm, the main idea is to find a condition that eliminates the longest edge between every triangle formed by the node u and two of its neighbors. In order for a pair of nodes $v, w \in N(u)$ to be selected both in the RNG of u, their incoming angles must be separated by more than $\pi/3$, $\|\theta_v - \theta_w\| > \pi/3$. Each node in the RNG generates a cone with $2\pi/3$ degrees of span, equally partitioned around the incoming angle of the neighbor node. Ideally, and assuming that all nodes are in the outer limit of the transmission range, a minimum of three nodes and a maximum of five nodes are required to cover the entire area of the beacon node, as shown in Figure 7.13.

All nodes in the DistRNG start using their minimum transmission power. They also define the variable $\Theta = \{zero\}$, which represents the area of a node that is being covered by the neighbors in the RNG. The protocol then proceeds as follows:

1. The transmission range is increased until the next closest neighbor v is found. If node v is not in the covered area, the edge (u, v) will be added to the RNG. If more than one uncovered nodes are found in this step, all of them are added to the RNG.
2. The angle of the new added node θ_v is used to update the total covered area Θ, by executing $\Theta \bigcup Cone_v$.
3. Steps 1 and 2 are repeated until $\Theta = 2\pi$, or the maximum power transmission is reached.

The main difference between DistRNG and CBTC is that DistRNG includes uncovered nodes in the set of selected neighbors instead of removing redundant nodes

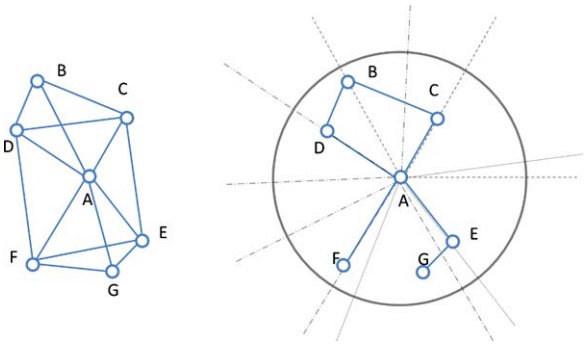


Fig. 7.14 Example of execution of the DistRNG on a simple topology.

after a broad selection process. This characteristic makes DistRNG a one-phase protocol, which definitely saves additional processing time while accomplishing the same goals of CBTC. Similar to the CBTC protocol, DistRNG is a distributed protocol that bounds the node degree to a value equal or less than 6, preserves network connectivity, utilizes local information, and generates bidirectional links. An example of the DistRNG protocol is shown in Figure 7.14.

Another directional-based protocol, which considers node mobility, is the **Angular Topology Control with Directional Antennas (Di-ATC) protocol** proposed in [40]. In order to avoid the overhead of keeping track of every single neighbor, each node only keeps track of the subset of neighbors that covers all its communication area, producing a similar structure as the one presented by the DistRNG protocol. Also, the algorithm includes a user-defined variable, d, that limits the neighborhood size. In this protocol, each node only polls the position of this subset of nodes, which number is tightly bounded, ≈ 6. A node u adds a new neighbor w to its set if any of the following conditions hold:

1. If $d_u < d$, where d_u is the node degree of u.
2. If $d_u = d$ and u does not have neighbors on a section of size $\Theta = \lceil 360/(d+1) \rceil$ around the position of node w.
3. If $d_u = d$ and u has at least one neighbor z on the area of coverage of w, but $d_w < d_z$. In this case, node u eliminates the link with node z.

Neighbor nodes update their positions permanently in order to know their new location (direction and distance). If the central node detects that one of its selected neighbors is not answering after a series of polling messages, it assumes that the link is broken and tries to add new neighbors to replace the lost one and complete the coverage area again. The Di-ATC protocol also provides a bounded node degree equal or less than d, or until the communication area is covered, which we know, it normally needs between 3 and 6 neighbors. Similarly, Di-ATC is distributed, provides network connectivity, uses local information, and generates bidirectional links. In addition, its update procedure is designed to handle node mobility.

7.5.3 Neighbor-Based Techniques

In previous techniques the topology construction algorithm requires extra information from the neighbor nodes other than their own presence, such as accurate Cartesian coordinate (bi- or tri-dimensional location) or polar coordinate (distance and angle). However, localization information is not always available or it could be very expensive to obtain. For example, location from GPS-enabled nodes can only be obtained in places where there is direct access to the satellite signals. Other localization techniques, like ultrasonic or ultrawide band-based, not only need a localization protocol on top of the topology construction protocol, but could also increase the communications overhead, as their range is very small compared with the radio coverage. In the case of polar coordinates, the use of directional antennae increases the price and complexity of the wireless devices.

Neighbor-based techniques overcome these problems, as they assume that nodes only need to have the ability to determine the amount of neighbors, change their transmission power and, in some cases, calculate the distance between nodes.

The main idea of these algorithms is to produce a connected topology by connecting each node with the smallest necessary set of neighbors, and with the minimum transmission power possible. Given that the nodes do not posses accurate location information, their decisions depend mostly on the probability of selecting the appropriate neighbors, the ones that would extend the network as far as possible. Under the assumption that the nodes are either uniformly or Poisson distributed, some properties have been found in connected topologies that define a bounded minimum appropriate size of the neighborhoods of a single node that *w.h.p.* would create a connected topology. As a result, most neighbor-based protocols for topology construction are based on the creation of a **K-neighbor graph**.

A K-neighbor graph is formally defined as in Equation 7.8 below:

$$G_k = (N, E_k), \text{ where } E_k = \{(u, v) \in E, u, v \in V \mid d(u, v) \leq d_k(u)\} \qquad (7.8)$$

where $d_k(u)$ is the distance from the k^{th} closest neighbor of u. In other words, each node will adapt its transmission range in order to have a direct link to its closest k neighbors. The main question of this approach is how to select the optimal k such that it will produce a connected topology.

The definition of the minimum number of neighbors k that each node must have in order to preserve connectivity has been a well-studied problem. Most commonly used numbers for this parameter are between 6 and 8, or an average of 3 neighbors, as presented in [65, 85, 51]. Using the experiments presented in Section 7.2 to determine the giant component and the average node degree, we found the following practical results:

- In order to have a connected topology with probability 1, the topology with 10 nodes showed an average neighborhood of 7, and the topologies with 100 and 1000 nodes showed an average neighborhood of 20 and 27, respectively.
- To have a connected topology with 90% probability, the topology with 10 nodes showed an average neighborhood of 5, the topology with 100 nodes showed an

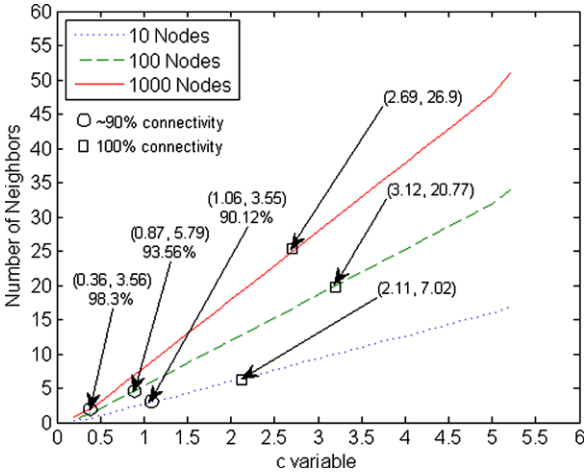


Fig. 7.15 Experimental value of c for different topologies.

average neighborhood of 10, and the topology 1000 nodes showed an average neighborhood of 9.

Recently, in [138], it is demonstrated that to produce a connected topology, each node should be connected to $\Theta(log\, n)$ nearest neighbors. Assuming a topology in $\Re^2$ where nodes are uniformly distributed and independent from each other, using a parameter $k = c\, log\, n$ where $0.074 < c < 5.1774 + \varepsilon$, for large values of n and ε a small positive constant, every node will be connected with its closest k neighbors. The authors show that, as n increases, the extreme values of c produce asymptotic trends to disconnection with probability 1, and connection with probability 1, in the lower and upper bound, respectively.

It is important to clarify that on the actual proof of connectivity, the authors define that there exists a link (i, j) in E_k, $i, j \in V$, if i is one of the k^{th} closest neighbors of j, or if j is one of the k^{th} closest neighbor of i. This assumption suggests that the two nodes may be connected with a single unidirectional link. In addition, it is worth noticing that this approach does not guarantee a *worst-case* connected topology. In real scenarios, having a common transmission range that satisfies connectivity only *w.h.p.* does not guarantee a connected topology in all possible scenarios. Further, the existence of unidirectional links might create a semi-partition of the network, because if a bottleneck link happens to be a unidirectional link, the subgraph on the destination side of the link will not be able to communicate with the subgraph on the side of the sender node. In general, the only way to completely guarantee connectivity in a worst-case scenario, under the homogeneous assumption, is by defining $k = n - 1$, which will produce a topology similar to the original MaxPowerGraph, assuming of course that it was connected from the beginning.

Although the authors of [138] state that the critical constant c may be close to one, it remains an open problem. Using the same simulation experiments utilized in Section 7.2, Figure 7.15 shows the curves produced by the formula $k = c\, log\, n$, for

three different values of n (10, 100, and 1000 nodes) and different values of c, and the average neighborhood size found in the experiments that produced connected topologies with 90% and 100% probability. As it can be seen, the value of the parameter $0.36 \leq c \leq 1.06$ provides a good estimate for having connected topologies 90% of the times, while values of $2.4 \leq c \leq 3.12$ provides a good estimate for having 100% connectivity, which agrees with the results shown in [138].

The **K-Neigh protocol**, introduced in [11], pretends to build a K-neighbor graph with exactly the same characteristics as the theoretical definition: each node will be connected with the closest k neighbors. In this protocol nodes are assumed to be able to calculate the distance between them and their neighbors.

The protocol starts with each node $u \in V$ transmitting an initial message at full power. Every node $v \in N_{MaxTx}(u)$ answers back. Then node u calculates the distance to each neighbor from which it received an answer and sorts them in the neighbors list L_u. After that, node u selects the closest k neighbors and sends a message at full power in order to share the new neighborhood list. Once node u has received the lists L_v from all its neighbors, it calculates the list of symmetric neighbors LS_u, and according to this list, node u adapts its transmission power to reach the farthest node in LS_u.

K-Neigh has several advantages. First, it is a distributed protocol. Second, it is very simple. Third, it has a low message complexity, $2n$. And fourth, it also bounds the node degree to equal or less than k. However, just as in the theoretical approach, K-Neigh does not guarantee a *worst-case* connected topology.

The same authors of K-Neigh proposed a similar version of the protocol called the **K-NeighLev protocol** [10]. This algorithm does not require the nodes to calculate the distance between them. Instead, K-NeighLev uses an incremental neighborhood discovery procedure until the size of the neighborhood is less than or equal to k. The authors justify this version based on the drawbacks of range estimation techniques: The *Receive Signal Strength Indicator (RSSI)* may not be very accurate in many cases because the metric is sensible to external disturbances; *Time of Arrival (TOA)* needs synchronization between the nodes (extra overhead for a synchronization protocol); and *Time difference of Arrival (TDOA)* needs two different ways to measure distance to implement the comparison (extra hardware and energy).

In the K-NeighLev protocol each node is assumed to have several levels of transmission power that define the size of the communication range. In addition, symmetry in communication between nodes is assumed, i.e., if node x can see node y using level l, then y can see node x with the same level l. Each node keeps a list of all nodes that are unidirectional neighbors (all nodes that can see the node, but cannot be reached with their current level) and all the bidirectional nodes (both nodes can see each other with their current levels).

The process of neighborhood construction starts with each node u sending a *beacon* packet using the minimum transmission level *minL* to discover the minimum neighborhood with symmetric links. When a node $v \in N_{minL}(u)$ receives a *beacon* packet from u, it includes u in its non-symmetric neighbor list. If the level used to send the *beacon* is less than or equal to the current level used by v, then u is considered as a symmetric neighbor by v.

Each node waits a certain amount of time T_0 called Stabilization Time, in which a node is supposed to receive *beacon* messages from all its neighbors. If after T_0 units of time the number of symmetric neighbors of u is less than k, u increases its transmission level $L_{current} = L_{previous} + 1$ and sends *help* messages, inviting other nodes $w \in N_{Lcurrent}(u)$ to become symmetric neighbors by increasing their own transmission level to $L_{current}(u)$. This process repeats until node u finds at least k neighbors or node u reaches its maximum transmission level.

When a node w receives a *help* message from u, it checks if u is a non-symmetric neighbor or if it is a completely new neighbor. In the first case, w executes the *stepwise_increase* procedure. This procedure increases the transmission level of w until it reaches $L_{current}(u)$, transforming the link between u and w in a symmetric link. On each step of the process, w sends *beacon* messages at every level announcing to other nodes its new level. Also, w checks if on the current asymmetric neighbor list there are nodes that become symmetric using the new increased level, adding them to the list of symmetric neighbors.

In the second case, i.e., if node u is a new neighbor of w, the *help* message received by w is initially assumed as a *beacon*, and then, if the current level of w is less than the one used in the *help* message, w executes the *stepwise_increase* procedure. If u is already a symmetric node of w, the *help* message is ignored.

An improved version of the K-NeighLev algorithm, called the unselfish K-NeighLev, or K-NeighLevU, was presented in [10]. This new protocol considers the following scenarios for optimization:

- If node u has less than k neighbors even at the last transmission level, it can decrease its level up to the level of the farthest neighbor.
- When a node u sends a *help* message, all nodes on the next level automatically increase their transmission level in such a way that they all can reach node u. Assume that node u has $k - 1$ neighbors at level l and there are c nodes on the next level $l+1$. According to the procedure, after u sends a *help* message, c nodes increase their transmission level and node u ends up with $k + c - 1$ neighbors, i.e., $c - 1$ more neighbors than needed.
- If node u has less than k neighbors, instead of sending a *help* message and making all of the new neighbors increase their transmission level, node u can check on the asymmetric neighbors list to see if by increasing its own level it could reach enough of them to fulfill the requirement of k neighbors.

Even though this new version requires extra steps, it may produce a better configuration of the neighborhoods. Also, by avoiding the generalized and mandatory symmetry between the transmission levels of all the neighbors nodes, the optimized algorithm increases the probability of saving energy. For example, assume that node u has less than k neighbors at level $l - 1$. Node u sends an *inquiry* message to all neighbors in level i. Each neighbor node $w \in N_i(u)$ replies including their current energy level. Once node u gathers all the information, K-NeighLevU offers different lines of action like: (a) increase the sender's own range in order to increase the number of symmetric links; (b) send selective help message to a set of neighbors asking them to increase their ranges of communication; or (c) just send a regular

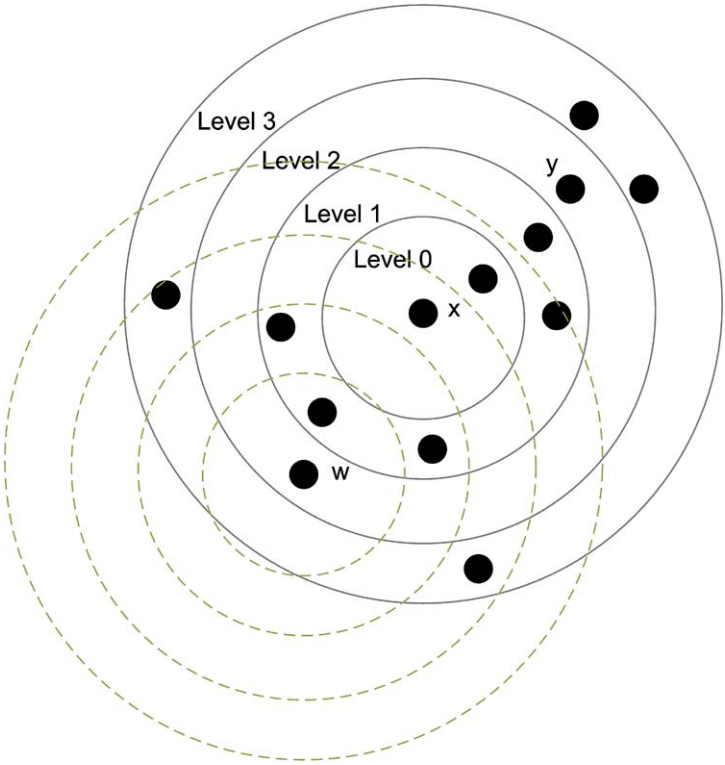


Fig. 7.16 Example topology for both selfish and unselfish K-NeighLev.

help message in which the sender asks every node on the next level to increase their transmission power. In general, the node should find the options that will increase the lesser amount of energy in the nodes of the network, but the authors say that the mechanism to predict energy increases is not feasible in practice and they apply heuristics to determine the best decision.

One example of the difference between the "selfish" and the "unselfish" K-NeighLev can be seen if applied on the example topology shown in Figure 7.16. If the original K-NeighLev algorithm were applied, assuming $k = 5$, nodes x and y would fulfill the requirement using a transmission power in level 1. However, node w needs to use up to level 2 in order to have at least 5 neighbors. When nodes x and z listen to the *help* message sent by w in level 2, they increase their transmission range to level 2. This is an unnecessary energy increase for x, as it already had its k neighbors. The ideal solution would be just to invite node z.

The K-NeighLev protocol is shown to have the following properties:

- Bounded node degree less than or equal to k, defined by the user.
- Distributed protocol.
- Only needs local information.

- K-NeighLevU implements some energy-efficiency changes compared to the original K-NeighLev version.

The **XTC protocol** [127] is an improved version of the K-Neigh protocol, which solves one of the main problem of the protocols based on the K-Neighborhood approach: the strictly bounded size of the neighborhood. This threshold is the main reason why connectivity in the worst-case scenario is not guaranteed.

The XTC protocol differs from the K-Neigh protocol in the following aspects:

- The sorting metric is link quality, not distance.
- The nodes exchange the complete list of neighbors, not a reduced list. This is important to detect the critical links that must be maintained.
- The size of the neighborhoods is not bounded; it depends on the number of nodes to be covered in the local range. This means that the XTC protocol guarantees connectivity even in the worst-case scenario.

XTC has the same message complexity than the K-Neigh protocol. Let us assume that each node u has an order function $\prec_u$ such that if u has two neighbors, v and w, and v has better link quality with u than w, then $v \prec_u w$. Based on the exchange of sorted lists of neighbors, each one checks the following condition on its neighbors:

For each v in $N(u)$ in decreasing order of link quality
if $(\exists w \in DN(u) \bigcup FN(u)$ such that $w \prec_v u)$
$DN(u) = DN(u) \bigcup \{v\}$
else $FN(u) = FN(u) \bigcup \{v\}$

where $FN(u)$ is the set of final neighbors of u, and $DN(u)$ is the set of discarded nodes of u.

In other words, the XTC protocol guarantees a connected topology by making each node u to verify that there is a link to every one of its discarded neighbors v through another node w, different than u, such that w has a stronger link with v than the one from u to v, and w has a stronger link with u than the one between u and v ($w \prec_v u$ and $w \prec_u v$). This guarantees that through w, node u can reach v in the case it is discarded, and also guarantees that if u is the best neighbor (or even the only one) to reach v without taking into account distance, then u will extend its area of coverage until it is able to cover v. This decision can be made locally because each node has the complete list of neighbors from its own neighborhood. The XTC protocol is simple, distributed, needs only local information, preserves the network connectivity, has low message complexity, $2n$, and has a bounded node degree of 6.

7.5.4 Routing-Based Techniques

The connectivity of a topology is one of the most important requirements of any topology construction protocol. One way to detect connectivity is by making sure that a route can be found from one node to every other node in the network. This is the main objective of the routing function: to build routing tables to route packets

from one node to all possible destinations. When all the nodes are included in the routing tables it implies that they can be reached, and the transmission range does not need to be adjusted. This is the main idea behind the routing-based techniques.

One of the most widely known topology construction mechanisms in this category is the **Common Power (COMPOW) protocol** presented in [84]. COMPOW's main objective is to find the minimum common transmission level that will produce a routing table with the same entries as the one found using the maximum transmission level. The authors affirm that there is a tight relation between the transmission level selection and the routing layer. For example, low transmission levels also imply low energy paths, low interference and good throughput. In fact, in [105], the author also shows how the minimum transmission power that achieves network connectivity is the optimal choice for increasing network capacity, reducing contention at the MAC layer, and reducing node energy consumption.

COMPOW assumes that the nodes have a certain small number of transmission power levels. At first, all the nodes start using the maximum energy level and execute any method for calculating an initial routing table. This routing table contains all nodes in the topology. Then, all nodes use the minimum transmission level and perform again the routing function. This process is repeated with transmission level increments until the routing table is equal compared with the one calculated using the maximum level. When this condition is reached, the network is connected and the nodes select their final transmission level. As it can be inferred, the COMPOW protocol is a practical implementation to solve the Critical Transmission Range problem.

The main problem of this protocol is the overhead produced by the several executions of the routing protocol and the need of global information. In order to reduce the overhead, the authors proposed a clustering algorithm to reduce the number of participating nodes in the routing tables.

7.6 Heterogeneous Topology Construction

The wide spectrum of possible applications where wireless sensor networks can be applied has increased the possibility of mixed networks, where devices of different types and characteristics co-exist and work in the same application. In this type of heterogeneous environment, it is very important to devise algorithms and mechanisms that will allow different devices to collaborate, each taking advantage of the abilities of the others.

Although topology control problems have been studied in the context of heterogeneous wireless sensor networks before, most existing mechanisms have focused on varying the nodes' transmission power based on the assumption that all the wireless devices have identical physical characteristics. As a result, topology control problems have been solved as range assignment problems, which not only neglect the heterogeneity of the network but also do not take advantage of the unique capabilities of different devices. This section presents three topology construction algorithms for heterogeneous wireless sensor networks: Directed LMST, Directed RNG, and the Residual Energy Aware Dynamic (READ) topology construction algorithm.

The **Directed LMST (DLMST)** and **Directed RNG (DRNG)** algorithms are proposed both in [71]. The motivation for heterogeneous topology control mechanisms is shown by the authors showing that several existing mechanisms cannot be directly applied to heterogeneous networks because they may render disconnected networks.

The proposed protocols, which are based on the LMST protocol and RNG algorithm, present the following characteristics:

- Connectivity and bidirectionality. If the original network is strongly connected and symmetric, the reduced topologies preserve these properties.
- Local information. Both protocols build the reduced topology based on locally collected information. Each node determines its transmission power based on local information and local decisions.
- Localization information. Both protocols assume that each node has the capability to know its location.
- Node degree. While the DLMST has a bounded node degree, the DRNG may be unbounded.

Both protocols work in three phases: the *Information Collection* phase, the *Topology Construction* phase, and the *Construction of Topology with Only Bidirectional Links* phase. This last phase is optional. During the first phase, each node collects ID and position information from its neighbors. During the topology construction phase, each node determines, according to each algorithm, which neighbors will be part of the reduced topology. The last phase makes sure that all links in the reduced topology are bidirectional. This is done either by enforcing unidirectional links to become bidirectional, e.g., increasing their transmission power, or deleting the unidirectional links, similar to what the same authors did in the LMST protocol for homogeneous wireless sensor networks.

Although these protocols preserve connectivity and bidirectionality, and have other nice features, they don't take full advantage of the heterogeneity of network devices. In the following, we describe the **Residual Energy Aware Dynamic (READ) topology construction algorithm** [145] for heterogeneous wireless sensor networks, a centralized algorithm where more powerful devices are set to have a more prominent role in the network connectivity to extend the lifetime of the network.[2]

READ considers the network model as a heterogeneous wireless network represented by a graph $G = (V(G), E(G), W(E), P(V), RE(V), \beta(V))$, where $V(G)$ is the set of nodes $V(G) = \{v_1, v_2, \ldots, v_n\}$, which are randomly deployed in a 2-dimensional network area, $W(E)$ is the *weighted cost* value associated to each edge in the graph, $E(G)$ is the set of edges, and $(P_{max\_v}, RE_v, \beta_v)$ are attributes associated with each node v. In the model, P_{max_v} is v's maximum transmission power in Watts, β_v is the sensitivity of v's receiver in dB, and RE_v is v's current residual energy in Joules, which decreases with v's activity from the initial value *INI_ENERGY*. READ assumes that node v is able to dynamically choose any

[2] A distributed version of READ can be found in [144].

power $P_v \in [0, P_{max\_v}]$ to transmit its packets, so that topology control can be instrumented.

Instead of using the Euclidean distance between the two communicating nodes to define the link cost, as in most homogeneous networks, READ introduces a *weighted cost* for each pair of nodes that considers both the energy for sending and receiving data and the current residual energy at each node. Assuming node u and v are within the neighbor set of each other, for one direction, $w_{u\to v}(e(u, v))$ represents the weighted cost for transmitting and receiving data from u to v, and $w_{v\to u}(e(u, v))$ represents the weighted cost for transmitting and receiving data from v to u. The weighted cost for transmitting and receiving data from node v to node u is calculated as:

$$w_{u\to v} = \frac{p_{u,v} \cdot t_{tx}}{RE_u} + \frac{p'_v \cdot t_{rx}}{RE_v} \tag{7.9}$$

where RE_u and RE_v are the residual energy at each node, $p_{u,v}$ is the minimum power for u to successfully send any packet to v, p'_v is the power for v to receive data, and t_{tx} and t_{rx} are the times for transmitting and receiving a given packet, respectively.

Similarly,

$$w_{v\to u} = \frac{p_{v,u} \cdot t_{tx}}{RE_v} + \frac{p'_u \cdot t_{rx}}{RE_u} \tag{7.10}$$

Hence, the weighted cost function reflects the proportion of energy required at both ends to perform a successful communication. Since READ considers bidirectional links only, communication costs in both directions are treated as a whole. Thus, the weighted cost is defined as:

$$w\big(e(u, v)\big) = w_{u\to v} + w_{v\to u} \tag{7.11}$$

READ has two phases: the Initialization phase and the Topology Construction phase. The Initialization phase consists of the following steps:

1. Construct the MaxPowerGraph, G_{max}, using the maximum transmission power of each transceiver. Note that G_{max} is bi-directional and is assumed to be strongly connected.
2. For each edge $e(u, v) \in E(G_{max})$, compute the weighted cost $w(e(u, v)) = w_{u\to v} + w_{v\to u}$.
3. Sort the set $E(G_{max})$ in increasing order of $w(e(u, v))$. To resolve ambiguities, the sorted edge sequence is renamed as E_{order}.

In the Topology Construction phase, the new network topology is built from the empty edge set $E(G_{READ})$. First, READ considers every node in the original network graph as an isolated component set G_i, i.e., $G_i = \{u_i\}$. Then, during the construction process, it merges two component sets at a time by adding one edge from $E(G_{max})$ to $E(G_{READ})$ until all the nodes have been connected and there is only one component set left. Thus, $G_{READ} = (E(G_{READ}), V(G_{max}))$ is the resulting topology which is also strongly connected and bidirectional. It is worth pointing out that the above algorithm can be applied k-times to preserve the inherited k-connectivity

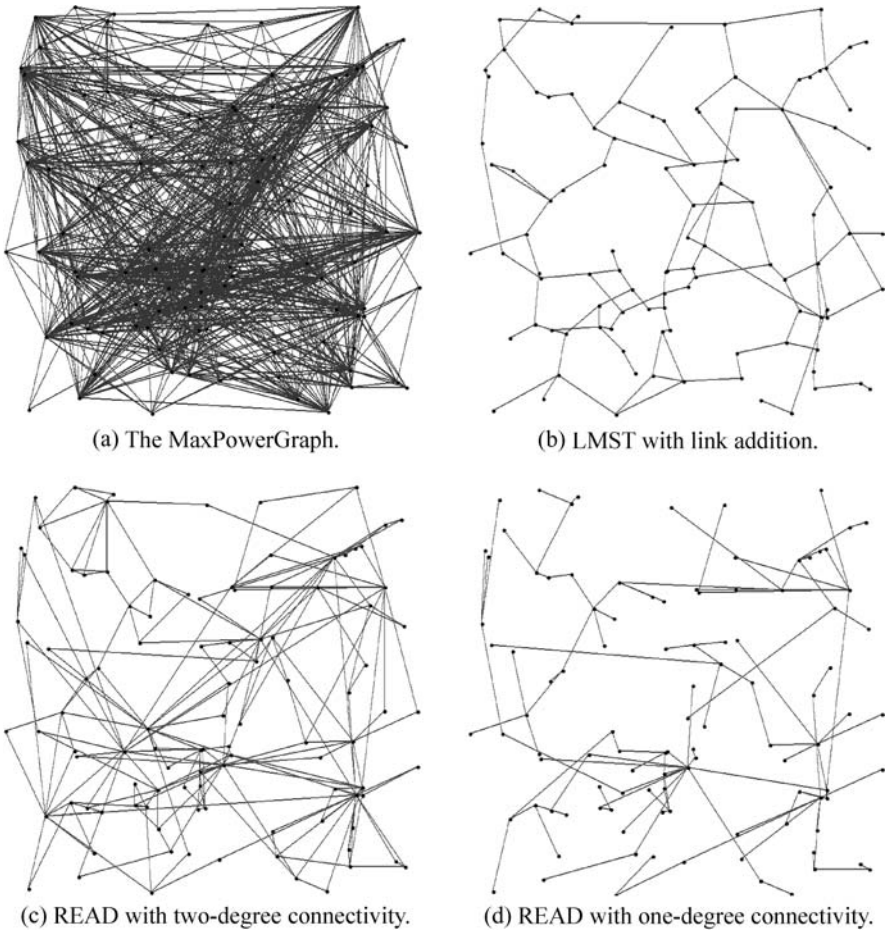
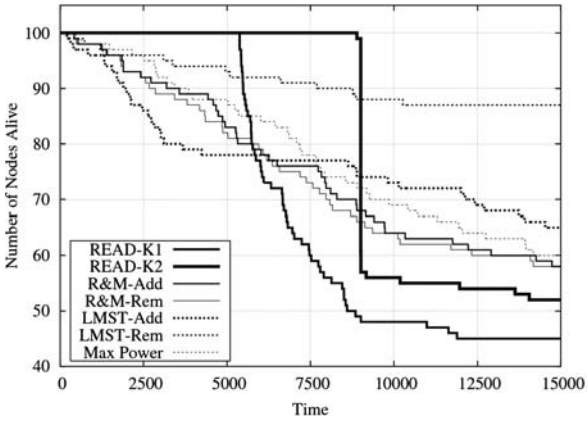


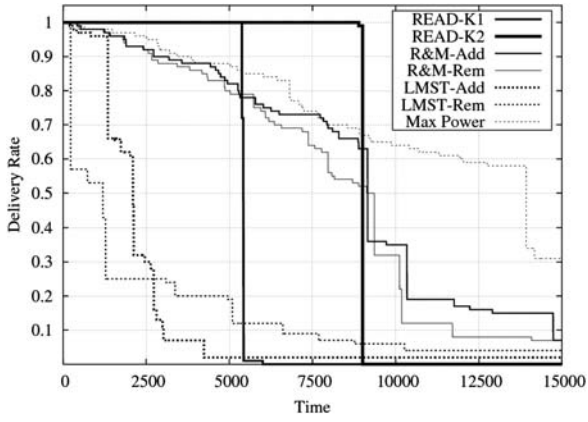
Fig. 7.17 Comparison of the effect of the READ algorithm on more powerful nodes compared with the LMST algorithm. Reproduced from [145] © 2007 IEEE.

of the MaxPowerGraph, and therefore, READ can have different versions according to the k-connectivity desired.

The READ algorithm was evaluated by simulations considering a heterogeneous wireless network with four types of devices: sensors, PDAs, Robots, and military devices. In the experiments, one hundred nodes were randomly distributed in a 1000 m $\times$ 1000 m network area, where 5% of them were powerful military devices, 10% of them were robots, 20% PDAs, and 65% of them were sensors. Each node utilized a uniform distribution to randomly draw its maximal transmission power in Watts, receiver sensitivity in dB, and initial energy in Joules from the pre-defined ranges $[P_L, P_H]$, $[\beta_L, \beta_H]$ and $[I\_E_L, I\_E_H]$ that characterize each type of device. The range of those parameters were determined based on the literature [23, 111, 115, 25]. The algorithm was run using $k-1$ and $k-2$ connectivity and com-



(a) Number of nodes alive.



(b) Packet delivery rate.

Fig. 7.18 The READ algorithm compared with LMST and R&M. Reproduced from [145] © 2007 IEEE.

pared with the LMST and R&M algorithms, both with their link addition and link removal versions [102, 73].

Figure 7.17 shows the effect that only the READ algorithm produces in the reduced topology, in which a few nodes (the more powerful ones) have a higher node degree than the rest of the nodes. This confirms that the READ algorithm is in fact assigning a more prominent role to those nodes that can afford it.

Figure 7.18(a) shows the effect of having some nodes with more prominent roles in the entire network. As it can be seen from the figure, the number of nodes alive of R&M-add and LMST-add start dropping linearly right after the simulation begins, while READ-K2 remains unchanged until $time = 9000$. This is very significant if we consider that at $time = 9000$ the network has sent around $100 \times 4 \times 9000 = 3.6 \times 10^6$ packets. This is even more significant if Figure 7.18(a) is seen along with Figure 7.18(b), which shows the packet delivery rate of the algorithms. As

it can be seen, while READ-K2 remains at 100% delivery rate, both LMST and R&M drop dramatically from the very beginning. At *time* = 9000, LMST-add has only less than 75 nodes alive and R&M has only around 70 nodes alive and their delivery rate dropped to 2% and 65%, respectively. One interesting observation is the case of LMST-rem, which maintains the number of nodes alive at a fairly slow decreasing rate compared to the rest of the algorithms. Although it may be thought of as the best algorithm, again, looking at Figure 7.18(b), it can be seen that the delivery rate is very low. The LMST and R&M algorithms present opposite trends. While LMST algorithms tend to keep more nodes alive than R&M algorithms, R&M provide better delivery rates than LMST. This basically means that R&M keeps the network more connected than LMST, even with a fewer number of nodes. Another interesting observation is that, with the exception of READ and LMST-rem, the performance of the remaining algorithms in terms of the packet delivery rate is very similar to the performance achieved by using the MaxPowerGraph, meaning that although the algorithms reduce the topology, this reduction doesn't translate into a useful performance advantage. Therefore, it is probably better not to utilize the algorithms.

Chapter 8
Building Hierarchical Topologies

8.1 Introduction

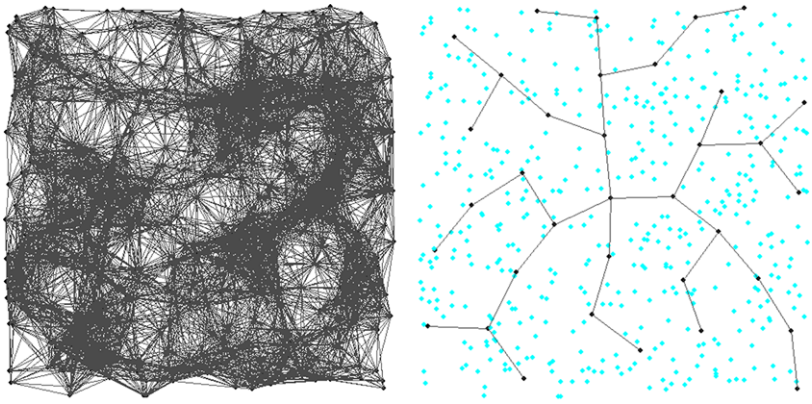
The previous chapter discusses how changing the transmission power of the nodes reduces the network topology, saves energy, and increases the lifetime of the network while preserving connectivity and coverage. However, this approach does not prevent the transmission of redundant information when several nodes are close to each other and may not simplify the network topology enough as to make wireless sensor networks scalable for large deployments. This chapter explains a different approach to topology construction, the hierarchical topology construction approach, which addresses the scalability problem and facilitates the aggregation of information for additional energy savings.

In the hierarchical topology construction approach, a communication hierarchy is created in which a reduced subset of the nodes is selected and given more responsibilities on behalf of a simplified and reduced functionality for the majority of the nodes. This approach has the potential to greatly simplify the network topology and the opportunity to save additional energy by assigning useful functions to the reduced subset of nodes, such as information aggregation and filtering, and routing and message forwarding.

One disadvantage of the hierarchical approach is that the selected subset of nodes will work more than their unselected neighbors, and will see their batteries drained sooner. Therefore, this approach must be accompanied by a good topology maintenance function that will rotate the role of the nodes with the final goal of spending their energy evenly and extending the network lifetime.

The classification of topology control mechanisms depicted in Figure 6.5 shows that hierarchical-based topology construction mechanisms can be classified as backbone-based, adaptive, and cluster-based. In the following sections, these categories along with the most important algorithms and protocols available in the literature are described and explained.

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,



(a) MaxPower Graph. (b) Reduced topology.

Fig. 8.1 Topology control by reducing the number of active nodes and the creation of a network backbone.

8.2 Backbone-Based Techniques

The main goal of the communication backbone approach is to find a subset of nodes that will guarantee connectivity and communication coverage throughout the deployment area while allowing every other node in the network to reach at least one node on this backbone in a direct way. It is important to mention that even though one common assumption of this approach is to have the nodes transmitting at full power, they could also reduce their transmission range to reach all their direct neighbors, so the approaches presented in Chapter 7 and this one are not mutually exclusive. Figure 8.1(b) shows an example of this approach in which 35 nodes out of 500 nodes in the network were selected to build the backbone. Also notice that these nodes provide complete network coverage.

A communication backbone can be created solving a widely known mathematical problem: the **Connected Dominating Set (CDS) problem**. A **Dominating Set (DS)** is a set of nodes $D \subset V$ in a graph, in which all other nodes that do not belong to the subset have a link to at least one node in the set. In the special case of the CDS, these nodes are connected. Mathematically, the formal definition of a DS is expressed as in Equation 8.1. Of course, the smaller the dominating set the better, therefore, finding the **Minimum Dominating Set (MDS)** and the **Minimum Connected Dominating Set (MCDS)** is of interest. These problems has been shown to be NP-hard in [39, 125], therefore, heuristics are needed for practical purposes.

$$D = \left(\forall v \in V : v \in D \vee \exists d \in D : (v, d) \in E\right) \tag{8.1}$$

In the following, we explain three methods used to create a connected dominating set: growing a tree, connecting independent sets, and pruning-based techniques.

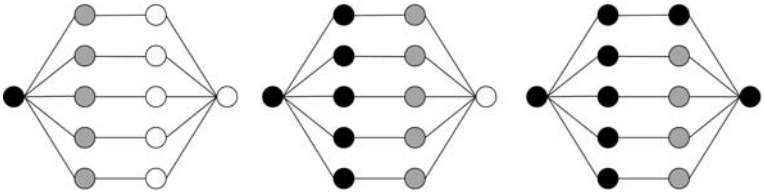


Fig. 8.2 Growing a tree with 1-hop neighbor information.

8.2.1 Growing a Tree

A CDS can be built growing a tree. An easy way to illustrate the tree construction process is by comparing it with Prim's algorithm [99], which finds the minimum spanning tree of a graph. The process works from the set of nodes that are part of the tree at time t. Between t and $t + 1$, based on certain parameters, the nodes in the set evaluate all adjacent nodes in order to extend the tree. The process continues until all nodes on the graph are evaluated. Of course not every node will be selected to be part of the tree, and those which were not selected will go to sleep until a new tree is requested. This procedure guarantees that the extended tree at $t + 1$ is still connected since the new selected nodes are always neighbors from at least one node of the tree set. In general, the process usually starts on a single node, the sink node, but if there are more than one sink node, then several trees may be built in parallel. This technique has characteristics that are desirable for routing protocols, like an organized structure that allows implementing routing based on search, or hierarchical addressing.

In [44], the authors present a coloring-based growing a tree approach to create a CDS on a graph. At the beginning, all nodes are marked as White, and the algorithm starts at the node with higher degree. This node is marked Black, and marks all its neighbors Gray. Gray nodes are inspected in order to calculate their "yield", or the number of White nodes that they would add to the tree. The Gray nodes that include more unmarked nodes on each iteration will be included on the tree by marking them as Black nodes. However, using the topology depicted in Figure 8.2, the authors showed that this approach is ineffective if Gray nodes looked at their own neighbors only. As it can be seen, the algorithm creates a tree with $d + 2$ nodes, where d is the node degree of the initial node, instead of the expected tree with only four nodes. The authors also show that the implementation of this algorithm runs in $O(m)$ steps, where m is the number of edges in the original graph.

In order to solve this problem, the authors proposed a modification of the algorithm, consisting of scanning pairs of nodes in a 2-hop manner. A Gray node and a White node will be marked as Black if their join contribution is the greatest on the iteration. First, the algorithm marks a Gray node Black, which makes all its neighbors Gray. Then, one of the Gray nodes is also colored Black, which makes its neighbors Gray. Here, the yield is given by the total number of Gray colored nodes. Finally, the pair of nodes with the highest yield is the one selected as part of the tree. This "look ahead" greedy solution not only produces the expected four-node tree but also

let the authors prove that the procedure produces a connected dominating set with at most $2(1 + H(\Delta)) \times |OPT_{DS}|$ nodes, where Δ is the maximum degree of the graph, $H(\cdot)$ is the harmonic function $H(k) = \sum_{i=1}^{k} 1/i \leq ln\, k + 1$, and $|OPT_{DS}|$ is the set of nodes in an optimal dominating set. The authors show that the implementation of this modified greedy algorithm can be run in $O(nm)$ steps, where n is the set of nodes and m the set of edges in the original graph.

A distributed version of this technique is presented in [27] with an approximation of at most $2H(\Delta)$ nodes than the optimal solution, and $O(|C|(\Delta + |C|))$ time and $O(n|C|)$ message complexity, where C is the dominating set produced by the algorithm.

The **A3 protocol** proposed in [128] is an example of a distributed implementation of a growing a tree algorithm. A3 builds a non-optimal connected dominating set over an originally connected graph considering the remaining energy in the nodes and the distance between them. The tree is built using four types of messages: *Hello Message*, *Parent Recognition Message*, *Children Recognition Message*, and *Sleeping Message*. The CDS building process is started by a predefined node that might be the sink, right after the nodes are deployed. The sink, node A in Figure 8.3a, starts the protocol by sending an initial *Hello Message*. This message will allow the neighbors of A to know their "parent". In Figure 8.3a, nodes B, C, D, and E will receive the message. Nodes F and G are out of reach from node A. If the node that receives the message has not been covered by another node, it sets its state as covered, adopts the sender as its "parent node", and answers back with a *Parent Recognition Message*, as shown in Figure 8.3b. This message also includes a selection metric (explained later) that is calculated based on the signal strength of the received *Hello Message* and the remaining energy in the node. The metric will be used later by the parent node to sort the candidates. If the receiver has been already covered by another node, it ignores the *Hello Message*.

The parent node waits a certain amount of time to receive the answers from its neighbors. Each answer (metric) is stored in a list of candidates. Once this timeout expires, the parent node sorts the list in decreasing order according to the selection metric. The parent node then broadcasts a *Children Recognition Message* that includes the complete sorted list to all its candidates. In Figure 8.3c, node A sends the sorted list to nodes B, C, D, and E. Once the candidate nodes receive the list, they set a timeout period proportional to their position on the candidate list. During that time nodes wait for *Sleeping Messages* from their brothers. If a node receives a *Sleeping Message* during the timeout period, it turns itself off, meaning that one of its brothers is better qualified to become part of the tree. Based on this scheme, the best node according to the metric will send a *Sleeping Messages* first, blocking any other node in its range. Therefore, only the other candidate nodes outside its area of coverage have the opportunity to start their own generation process. For example, in Figure 8.3d, node D received a *Sleeping Message* from E before its timer expired, so it turned off. Otherwise, it sends a *Sleeping Message* to turns its brothers off. At that time, this particular node becomes a new "parent node" and starts its own process of looking for candidates. Finally, if a parent node does not receive any *Parent Recog-*

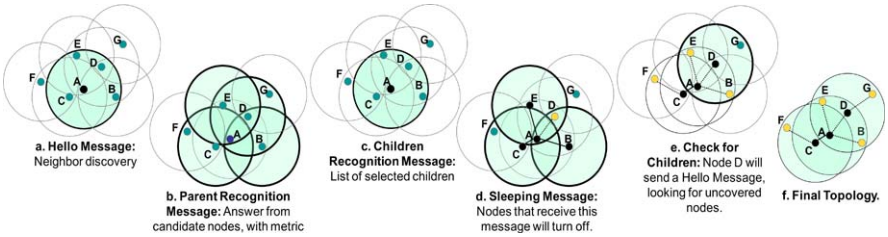


Fig. 8.3 Example of the execution of the A3 algorithm.

nition Message from its neighbors, it also turns off, such as the case of nodes E and B in the final topology, as shown in Figure 8.3e, given that they have no children.

A3's selection metric is given by Equation 8.2, which gives priority to those nodes with higher energy and which are farther away from the parent node, expecting to build a tree with fewer nodes and better coverage.

$$Mx, y = W_E \cdot \frac{E_x}{E_{max}} + W_D \cdot \left(\frac{RSSI_y}{RSSI^*}\right) \tag{8.2}$$

where x is a candidate node, y is a node inviting x, W_E is the weight for the remaining energy in the node, W_D is the weight for the distance from the parent node, E_x is the remaining energy in node x, $RSSI_y$ is the received signal strength from the parent node, and $RSSI^*$ is the minimum RSSI to ensure connectivity, which is given by the sensitivity of the receiver. Equation 8.2 produces a value between 0 and 1 that is assigned to each neighbor in the process of selecting the new nodes in the tree; the higher the value of the metric, the higher the priority.

A3 presents the following advantages: (a) A3 is very scalable, as it only needs local information and operates in a completely distributed manner; (b) A3 does not need location information; no GPS or any localization mechanism is necessary; (c) A3 requires no synchronization scheme thanks to the ordered sequence of the tree creation; (d) A3 is simple, and presents low computational complexity; and (e) A3 is very energy-efficient. First, the CDS tree building process is done in only one phase and the node selection process avoids the use of node contest or two-hop information queries, which reduces the amount of overhead. Second, the small number of active nodes after the topology construction reduces the number of collisions considerably. Finally, A3 has a low and linearly bounded message complexity of $4n$, where n is the number of nodes. This last advantage is very important since it allows for the algorithm to be run many times as part of the topology control iterative cycle with very low energy cost.

8.2.2 Connecting Independent Sets

Another approach to building a CDS tree is to create independent sets first, and connect the sets later. During the first phase, the idea is to form a **Maximal Independent Set (MIS)**. An **Independent Set (IS)** of graph G is a subset V where no

two nodes in V have an edge. An MIS is an independent set that is not a subset of any other independent set, i.e., an MIS of G is an IS that cannot include more nodes in V. Therefore, an MIS is a Dominating Set, which may not be connected. During the second phase, these algorithms find a way to connect the DS with the minimum number of nodes, and therefore, form the CDS tree.

The **Energy Efficient Connected Dominating Set (EECDS)** algorithm is an important example in this category. Proposed in [143], the EECDS algorithm creates a maximal independent set in the first phase, and then selects gateway nodes to connect the independent sets during the second phase.

EECDS also uses a coloring approach to build the MIS. The EECDS algorithm begins with all nodes being White. An initiator node elects itself as part of the MIS coloring itself Black and sending a Black message to announce its neighbors that it is part of the MIS. Upon receiving this message, each White neighbor colors itself as Gray and sends a Gray message to notify its own White neighbors that it has been converted to Gray. Therefore, all White nodes receiving a Gray message are neighbors of a node that does not belong to the MIS. These nodes need to compete to become Black nodes. The competition consists of sending an *Inquiry* message to its neighbors to know about their state and weights and wait for their responses for a specific amount of time. If during this time, it does not receive any Black message in response, and it has the highest weight, it becomes a Black node, and the process starts again. Otherwise, it stays as a White node. The weight is a metric calculated by each node based on the battery power and effective node degree. The authors show that the set of Black nodes produced by the above algorithm forms a MIS.

The goal of the second part of the algorithm is to form a CDS using nodes that do not belong to the MIS. These nodes, called *connectors*, are selected in a greedy manner by MIS nodes using three types of messages. A non-MIS node that becomes part of the CDS sends a Blue message to notify its neighbors. MIS nodes send *Invite* messages to non-MIS nodes to invite them to be connectors. In response to invite messages, non-MIS calculate their weights and send *Update* messages. Finally, the non-MIS node with the highest weight becomes part of the CDS. The authors prove that phase two of EECDS builds a CDS.

The main disadvantage of the EECDS algorithm is its message complexity. In both phases of the algorithm, competition is used to determine the best candidates to be included in the independent sets and the final tree. This process is very costly in terms of message overhead because each node has to consult its neighbors for their status in order to calculate its own metric. This large overhead is particularly detrimental in dense networks because of the network congestion and collisions that it generates. The authors show that the message complexity of the EECDS algorithm is $O(n)$, as during each of the two phases each node at most sends out one message. The time complexity of the algorithm is given by the construction of the MIS, which has a $O(n)$ worst time complexity. The authors also show that EECDS has an approximation factor of not higher than 7.6, i.e., the EECDS produces a CDS with at most 7.6 times the number of nodes given by the optimal solution, which is the Minimum CDS. Finally, the authors propose a backbone recalculation procedure to switch the backbone when the minimal energy of the current one decreases by 50%.

This procedure, unfortunately, is not well-explained in the paper and it is unknown whether it needs global information or it can be triggered in a distributed manner. Further, the claim that the recalculation procedure balances the energy consumption of the nodes is not supported.

The **Distance-2 (D2) Coloring algorithm** [88] is another approach in this category. The authors present the idea of building a Well Connected Dominating Set (WCDS), which is defined as a CDS with low size, low degree and low stretch properties. The WCDS is first solved in a centralized manner in two phases. In the first phase the algorithm gets the unit disk graph $G = (V, E)$ and computes an MIS. In the second phase the algorithm connects nodes x and y in the MIS. The path from x to y is the shortest path considering all nodes in the 3-hop neighborhood of x.

The distributed version of the algorithm is performed in three phases. The first phase utilizes the D2-coloring technique, which assigns one color to each vertex x and makes sure that all vertices within 2 hops away from x are assigned different colors. This D2-coloring assignment is motivated by the fact that nodes with the same color can transmit simultaneously without collisions. One of the contributions of the authors is solving the problem of minimizing the number of colors, which has been proved NP-hard in [109] in a centralized setting, in a distributed manner using unit disk graphs with running time $O(\Delta log^2 n)$. During this phase nodes pick colors and exchange messages to decide whether to change their colors or make them permanent. The second phase involves building a MIS. This phase is built in an iterative manner in time slots and assumes that phase one has taken place, i.e., no two nodes have the same color in node x's 2 hop neighborhood. During slot i all nodes with color i attempt to join the MIS. A node joins the MIS if none of its neighbors belong to the MIS. After node x joins the MIS, it broadcasts a message to its neighbors indicating that it joined the MIS. Finally, the third phase connects the MIS. During this phase, nodes exchange several messages to know their 3-hop neighborhood and all the paths of length at most three between each node and its 3-hop neighbors. During this process the nodes collect enough information to connect the nodes as in the centralized algorithm. In their work, the authors prove that after all these phases the total number of messages transmitted by the algorithm is at most $O(n\, log^2 n)$.

Another classic algorithm that uses a CDS approach is **SPAN** [20]. SPAN introduces the concept of the *coordinator*, a node that, based on local decisions, decides to stay awake and be part of the backbone. Periodically, every node on the graph explores its neighborhood. A node decides to become a coordinator on behalf of two neighbors if it finds that those two neighbor nodes cannot communicate with each other directly or through any other coordinator. If this is the case, the node announces its decision of becoming a coordinator, which blocks any intention of another node on its neighborhood to become part of the backbone. This selection policy not only maintains connectivity but also preserves network capacity.

SPAN also includes a mechanism to rotate the role of coordinators and therefore spends the nodes' energy more evenly. Each coordinator periodically exchanges massages with its neighborhood and withdraws itself from its role if it finds that all pairs of neighbors can communicate either directly or via another coordinator. In

addition, coordinators remove themselves from the backbone after they have been coordinators for some period of time, but only if they find other nodes in their neighborhoods that can make all pair of neighbors to communicate with each other, i.e., if they can find another coordinator from their neighbors. The rotation period of time is based on the amount of energy available, and the number of pairs of neighbors that the node can connect together. SPAN presents an $O(n)$ message and computational complexity.

A modified version of this algorithm that includes the use of directional antennas and presents the same message and computational complexity of SPAN is presented in [67]. Other algorithms that belong to this category can be found in [88], [119], and [44].

8.2.3 Pruning-Based Techniques

In the approach described in Section 8.2.1, the algorithms start with a reduced non-connected topology and then add nodes to connect it. In the Pruning-based techniques approach, the algorithms calculate a topology that guarantees connectivity by including most of the nodes, and then unnecessary nodes are pruned out. This approach is used in the Connected Dominating Set under Rule K algorithm proposed in [133, 131] and the Extended Connected Dominating sets introduced in [130].

The **Connected Dominating Set under Rule K (CDS-Rule-K) algorithm** proposed in [133, 131] is an example of a distributed algorithm under this category that uses local information. The algorithm works in two phases. The first phase involves creating an initial CDS tree using the following marking process [133]:

$$S = \left(\forall v \in S : x, y \in N_v, \neg\exists(x, y) \in E\right) \tag{8.3}$$

where N_v is the set of neighbors of v.

In other words, if node v has two neighbors that are not connected, it will include itself on the initial set. In this first phase, the nodes exchange HELLO messages in order to get to know their neighbors and exchange their neighbor lists. Once a node receives the lists from its neighbors, it intersects the lists with its own list of neighbors and counts the number of elements in both lists. If this number is less than the amount of neighbors in the node's own list, then this node will mark itself as part of the initial set.

For the second phase, the algorithm prunes unnecessary nodes applying one of the three pruning rules described in [131]. A temporarily marked node decides to unmark itself if it determines that all its neighbors are covered by marked nodes with higher priority, which might be given by the level of the node in the tree: lower level, higher priority. An initiator node requests the list of all marked nodes in its neighborhood with less or equal priority than itself. The neighbor nodes update their list of marked nodes and respond with the updated list. Then, based on the reply messages, the initiator checks if all its neighbors are covered by at least one active node different from itself. If this is the case, it unmarks itself and announces it, so the other nodes can update their metrics; otherwise, it marks itself as a permanent

element of the tree. The final tree is a pruned version of the initial one with all redundant nodes that were covered by other nodes with higher or equal priority removed. This is the first rule, called *Rule 1*. In Rule 1, a marked host can unmark itself if its neighbor set is covered by another marked host; that is, if all neighbors of a gateway are connected with each other via another gateway, it can relinquish its responsibility as a gateway.

Accordingly, in the *Rule 2* pruning approach, a marked host can unmark itself if its neighborhood is covered by two other directly connected marked hosts. Finally, the third and last rule, *Rule K*, generalizes the approach and builds a better (smaller) CDS relaxing the constraint on the number of marked hosts, i.e., *Rule K* unmarks gateways covered by k other gateways, where k can be any number.

As in the case of the EECDS algorithm, the node query and the unmarking announcement processes of the CDS-Rule-K mechanisms are the source of a great amount of messages. During the query process each node generates a query that is based on its own level and receives individual answers. During the unmarking announcement process, once a node decides to unmark itself, it must update all its neighbors to correct the metrics in all possible active processes. This is two-hop information that most of the nodes send once they get pruned from the tree. The CDS-Rule-K mechanisms has an $O(n^2)$ message complexity, $O(n^2)$ computational complexity, and an $O(1) \cdot DS_{opt}$ approximation.

In a later work [130], the notion of the **Extended Connected Dominating Set (ECDS)** is introduced, which is defined as the connected set where every node in the network is in the set, it has a neighbor in the set, or it has k 2-hop neighbors in the set. The idea of the ECDS is to form an even smaller tree that would therefore translate into more energy savings. Based on this idea, the authors modify several existing algorithms and create their "extended" versions. For example, the authors create the E-MCDS algorithm, the extended version of the MCDS algorithm described in [28], which is at the same time a modification of the "growing a tree" scheme proposed in [44] and described in Section 8.2.1. In addition, the authors proposed E-AWF, the extended version of the AWF algorithm proposed in [3] and the E-Rule-K algorithm, the extended version of the Rule K algorithm described above. In all these extended versions, the authors show the superiority of the extended concept in terms of final number of nodes in the tree.

8.3 Cluster-Based Techniques

The algorithms presented in Section 8.2 create a reduced structure that can be used as a backbone for communication, i.e., a reduced version of the complete graph that still offers complete connectivity and coverage of the deployment area. This section presents cluster-based techniques, which simplify the network topology even further creating groups of nodes (clusters) managed by special nodes called Clusterheads. Clusters are created by clustering techniques, which classify objects into different groups, or more precisely, partition the data set into subsets (clusters) so that each cluster shares some common feature, characteristic, or trait. Cluster-based

techniques also include algorithms to select the clusterheads, special nodes that not only serve as facilitators for communication on behalf of their cluster nodes but also can perform special tasks such as routing, data aggregation, scheduling, and others with the potential to save even more energy.

Both clustering techniques and clusterhead selection algorithms are very important in these topology construction techniques since they have to be implemented in a distributed fashion, consume the network energy evenly and efficiently, and introduce the least amount of extra overhead. For example, it is important to select the best node in a cluster as the clusterhead. In order to have a way to determine if a candidate is better than other, candidates must provide a selection metric, which will measure the benefit of selecting it as a clusterhead. If several nodes consider that they could become clusterheads because their metrics are high or are over a certain threshold, or just after a random decision, they must compete in order to determine the best. Competition usually means exchange of messages, which introduces extra overhead. Also, since clusterheads will perform additional functions compared with the rest of the nodes in the clusters that they represent, their energy will be drained at a faster pace. Therefore, additional mechanisms must be included to rotate the role of clusterhead. This rotation can be performed in many different ways, such as periodically, based on remaining energy, randomly, etc., with the ultimate goal of allowing the participation of all the nodes in a cluster as clusterheads and consuming their energy in an evenly manner.

One of the most widely known cluster-based techniques is the **Low Energy Adaptive Clustering Hierarchy (LEACH) protocol** presented in [47], which has already been introduced in Section 3.2.2, as part of the MAC layer, and in Section 4.2.2 as a cluster-based routing mechanism. In this section, we will not describe the protocol and its benefits again but focus on the clustering technique and the clusterhead selection algorithm.

In LEACH, time is divided in rounds. During the set up phase of each round, clusterheads are selected first and clusters organized afterwards. During the steady-state phase, normal network operation takes place. The clusterhead selection algorithm is executed during the set up phase of each round and guarantees that each node will become a clusterhead at some point during the network lifetime, i.e., LEACH rotates the role of clusterhead on a round by round basis in a fair manner so as to consume the nodes' energy evenly and therefore extend the network's lifetime. This selection mechanism is implemented by a threshold-based random procedure that includes the calculation of the probability of becoming a clusterhead as follows:

$$T(n) = \begin{cases} \frac{P}{1-P\times(r \bmod \frac{1}{P})} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases} \tag{8.4}$$

where $T(n)$ is the threshold for node n, P is the desired percentage of cluster heads, which is a network parameter decided a priori, r is the current round, and G is the set of nodes that have not been clusterheads in the last $\frac{1}{P}$ rounds. Once each node computes its threshold, it draws a random number between 0 and 1 to make the final decision. If the number is less than the calculated threshold $T(n)$, the node becomes

a clusterhead. It is easy to see that each node will become a clusterhead during in one of the $\frac{1}{P}$ rounds, as nodes already selected as clusterheads in one round cannot become clusterheads again, and the probability of becoming a clusterhead for the non-selected yet nodes increases with the number of rounds.

LEACH's clustering technique is based on the received signal strength from the clusterheads. Once a node becomes a clusterhead, it broadcasts an advertisement message to the rest of the nodes. Nodes measure the signal strength at which they received the advertisement message and decide to join the cluster with the largest value.

In LEACH clusterheads aggregate data received from their cluster nodes, send aggregated data directly to the sink on their behalf, and also act as a base station implementing a TDM-like MAC scheme and sending the transmission schedule that the nodes use to go back to sleep mode. On the down side, LEACH practical applicability is restricted to networks of maximum sizes given by the maximum transmission range of the clusterheads.

Another well-known cluster-based technique is **Hybrid Energy-Efficient Distributed (HEED) clustering**, a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks [142]. HEED addresses the limitation of LEACH allowing clusterheads to be more than one hop away from the sink node. It guarantees intercluster connectivity, and therefore the scalability problem of LEACH is solved by building a higher layer structure capable of routing data from cluster to cluster all the way to the sink. This higher layer structure can utilize any routing scheme, such as those included in Chapter 4, so that data finally reach the sink node.

As in LEACH, HEED selects clusterheads and organizes the clusters periodically. One option in HEED is the availability of nodes with several power levels, in which case nodes use a smaller transmission level, proportional to a square cell of size c, to create the cluster and increase spatial reuse, and a higher transmission range of size $(1 + \sqrt{5})c$ to guarantee intercluster connectivity. According to this, the intracluster transmission level determines the number of clusters in the network.

Since HEED's main goal is to extend the network lifetime, the clusterhead selection metric is primarily based on the residual energy of each node. Nodes initially calculate their probability of becoming a clusterhead, CH_{prob}, using Equation 8.5.

$$CH_{prob} = C_{prob} \times \frac{E_{residual}}{E_{max}} \tag{8.5}$$

where C_{prob} is a predetermined number of clusterheads in the network (as in LEACH), $E_{residual}$ is the estimated remaining residual energy in the node and E_{max} is a reference energy value corresponding to the amount of energy that the nodes have when fully charged. With this probability, the nodes elect to become part of the set of possible clusterheads, S_{CH}. Then, nodes choose their clusterheads from the set S_{CH} according to a cost function, which can be different according to the goals of the network designer or network operator. Cost may be based on the node degree of the nodes and the amount of power that all nodes in the cluster will con-

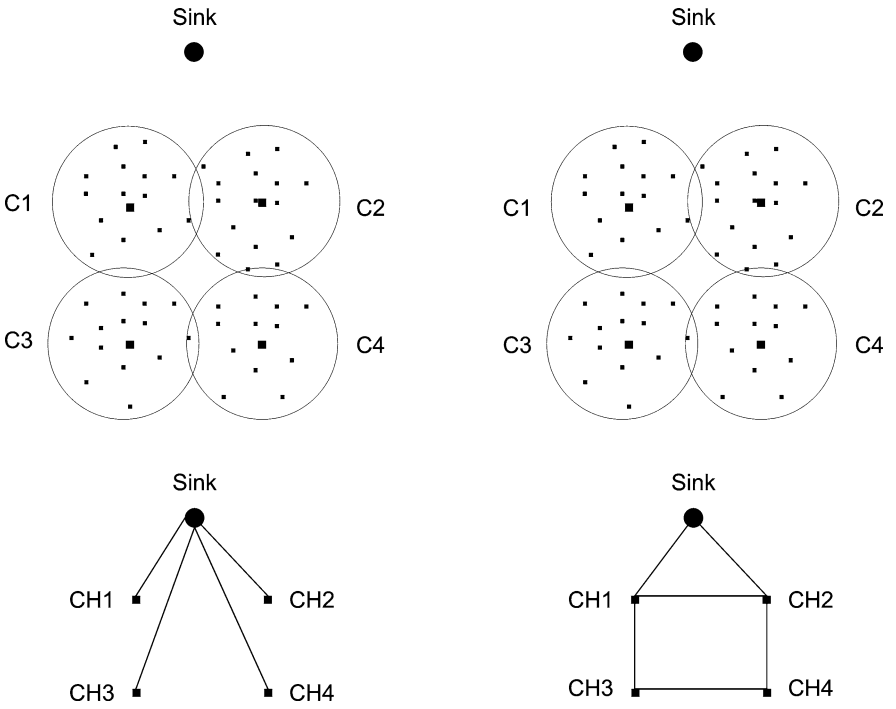


Fig. 8.4 The LEACH and HEED clustering protocols.

sume if a particular node is chosen as the clusterhead, if the goal is to distribute the load evenly; or it can be the inverse of the node degree if dense clusters are desired.

Figure 8.4 shows the LEACH and HEED protocols and the type of higher layer topology that each one may create. With the energy-based clustering metric and the intercluster communication capabilities, HEED can incorporate nodes with different capabilities and scale better than LEACH. Both LEACH and HEED have an $O(n)$ message and computational complexity.

8.4 Adaptive Techniques

Adaptive techniques are those mechanisms that build the reduced topology adjusting it over time and according to some conditions. Adaptive techniques for topology construction in wireless sensor networks are not very common in the literature thus far. One interesting feature of these techniques is that they perform both, topology construction and topology maintenance functions.

The **Adaptive Self-Configuring sEnsor Networks Topologies (ASCENT) mechanism** presented in [19] is perhaps the most relevant algorithm in this category. ASCENT is a measurement-based mechanism whereby nodes decide to participate in the reduced topology according to the packet loss rate that they experience. For

example, when a node measures a high packet loss rate, it requests additional nodes to join the network. Similarly, it does not participate in the data forwarding process if the packet loss rate is low, reducing collisions and saving energy.

ASCENT separates itself from many of the algorithms described so far in the sense that it is a measurement-based scheme addressing some issues not well studied by simulations or analytical means, such as the existence of asymmetrical channels in low power radios, nonisotropic areas of coverage, and nonmonotonic signal strength reductions with distance. In addition, ASCENT neither needs any localization mechanism nor depends on any routing protocol.

In ASCENT, nodes may be in any of four states: active, passive, test, and sleep. Initially, only some nodes are active and the rest remain passive, listening but not transmitting. Passive nodes are called by active nodes to help in the data forwarding process – join the network and become active – if current active nodes measure high packet loss rates. In other words, high packet loss rates are an indication of a poor topology, which lacks enough active nodes. When a passive node receives this help message, it may decide to join the network. Depending on certain parameters, the node may decide to enter the test state, where the node probes the network to make sure that its decision to join the network will actually improve connectivity. If so, the node becomes active and remains active until its battery is completely drained. Otherwise, the node goes back to the passive state. Once there, the node switches its state back and forth to and from the sleep state. While in the passive state, the node continues gathering information regarding the state of the network and therefore, it has the opportunity to go to the test state according to the measurements and parameters. While in the sleep state, the node turns its radio completely off for a predetermined amount of time.

ASCENT has several good properties. For example, it provides methods to dynamically adjust the values of the parameters it uses to make state change decisions, such as the loss threshold (LT), which defines the maximum data loss before active nodes ask for help, and the neighbor threshold (NT), which defines the average degree of network connectivity. Also, it is MAC and network layer independent. ASCENT is a software layer that sits between the Data Link Layer and the Network Layer that can work with any MAC and routing protocol. On the low side, ASCENT uses active nodes until they run out of energy. This may create network partitions earlier than necessary, limiting the lifetime and usefulness of the network, and because of the same reason, ASCENT does not balance the energy consumption and traffic load among the nodes in an evenly manner.

The **Minimum Power Configuration Protocol (MPCP)** and the **Minimum Active Subnet Protocol (MASP)** proposed in [134] are the distributed versions of two centralized mechanisms that consider power aware routing and sleep management in a joint optimization problem. By doing so, the schemes minimize the total transmission energy by a packet on its route toward the destination and reduce the idle energy by turning unnecessary nodes off.

The centralized versions of MPCP and MASP assume that all data sources are known in advance, and create a reduced optimized topology from all the given data sources to the sink node. These schemes are not very practical in wireless sensor

networks because data sources are triggered by asynchronous events not known a priori and because the established topology may not guarantee complete coverage. The distributed versions solve these problems including mechanisms to discover new sources and establish their optimal paths toward the destination. As such, the topology is adaptively changed including new nodes and removing old ones over time.

The implementation of MPCP is based on the Destination Sequenced Distance Vector (DSDV) [96] routing protocol for ad hoc networks, which is a distributed implementation of the Bellman–Ford shortest path algorithm. However, in MPCP the cost metric not only depends on the hop count but also considers the operational state and the data rates of the nodes. The only difference between MPCP and MASP is that MASP does not use the data rate in the link cost, which helps reducing the storage overhead of the routing table at each node and the network bandwidth used by route updates. One of the major drawbacks of these algorithms is that they don't balance the network load among the nodes, and therefore, some nodes may get their energy depleted faster than others, which at the same time may disconnect the network sooner than expected.

Chapter 9
Hybrid Approaches

9.1 Introduction

The techniques presented in previous chapters can be combined in order to develop a more complete and better solution to the topology construction problem. For example, clustering techniques might be used along with controlling the transmission power of each node to simplify the topology and solve the RA problem, saving additional energy. This chapter describes several hybrid approaches available in the literature.

9.2 Hybrid Techniques

An interesting hybrid technique, called **CLUSTERPOW**, is presented in [63] where the authors combine clustering, router-based, and power control techniques to solve the Range Assignment problem in networks with non-homogeneous deployments. CLUSTERPOW is a generalized version of the routing-based COMPOW technique described in Section 7.5.4. In COMPOW, the algorithm increases the transmission power of the nodes until the routing tables found with the current transmission level are equal to the routing tables found using the maximum power. COMPOW offers a good solution in a network of uniformly deployed nodes; however, in non-homogeneous deployments, COMPOW fails to select a more optimal transmission power. This problem is depicted in Figure 9.1, which shows one part of a wireless sensor network with non-homogeneous deployment. If the COMPOW protocol were used in a network with non-homogeneously deployed nodes, like the one in the figure, COMPOW would choose a non-optimal minimum transmission power equal to the power needed to connect the two clusters. On the other hand, if clustering techniques are jointly exercised with individual power control, only cluster connecting nodes will choose a higher transmission level. The COMPOW case is like solving

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

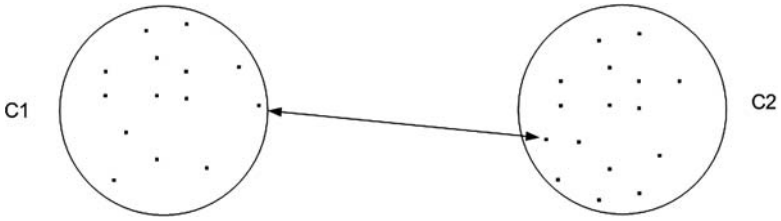


Fig. 9.1 The CLUSTERPOW protocol.

the CTR problem using a routing-based method whereas the CLUSTERPOW case is like solving the RA problem using routing- and cluster-based techniques.

As COMPOW, CLUSTERPOW nodes run several routing daemons, one for each available transmission power level, which build separate routing tables. As it can be inferred, COMPOW is a special case of CLUSTERPOW in the case of homogeneous deployments. One key difference between the two schemes is the implicit, adaptive, and distributed clustering technique with no clusterheads or gateways of CLUSTERPOW.

Another hybrid protocol is the **Topology Management by Priority Ordering (TMPO) protocol**, introduced in [6]. TMPO builds a hierarchical topology combining clustering techniques and dominating sets. TMPO uses clustering to build a minimal dominating set and then selects especial nodes called *doorways* and *gateways* to build a connected dominating set. TMPO finds a minimal dominating set selecting a set of clusterheads from the initial network. In TMPO, a node decides to become a clusterhead if it has the highest priority in its one-hop neighborhood, or if it has the highest priority in the one-hop neighborhood of one of its one-hop neighbors. The priority of the nodes consists of the identifiers of the node's neighbors, the present time, and the "willingness" value, which is a function of the remaining energy and the mobility of the node. As such, TMPO is more targeted toward mobile ad hoc networks than static wireless sensor networks. Since each node runs the same algorithm and the same data to determine its priority and the priority of its neighbors, each node knows which node should be the unique clusterhead. Upon selecting the clusterhead, a two level hierarchical network is established in which the clusterhead is at the higher level and the rest of the nodes in the neighborhood are at the lower level. TMPO is a time-slotted mechanism that rotates the role of clusterhead among nodes. At the end of each recomputation period T, nodes calculate the priorities and select a new clusterhead.

The authors of TMPO include two important proofs: first, they prove that the set of selected clusterheads is a dominating set, and second, that in a dominating set, the maximum distance between closest clusterheads is three. These two proofs are used by TMPO to select *doorways* and *gateways* and establish the connected dominating set. In order to connect the clusters, TMPO proceeds in two steps. In the first step, the algorithm selects doorways to bring two clusterheads one hop closer to each other. This is achieved by including doorways in the dominating set; since there are no more than three hops between two clusterheads, the algorithm selects

as a doorways the node with the highest priority on the shortest path between the clusterheads. In the second step, the algorithm connects clusterheads to clusterheads or clusterheads to doorways that are only two hops away by selecting the node with the highest priority between them. These nodes, which are also included in the dominating set, are called gateways, and complete the connected dominating set.

The **Topology and Energy Control Algorithm (TECA) for Wireless Sensor Networks** proposed in [17] is very similar to TMPO. TECA also utilizes a clustering technique to create independent sets and then include *bridges* to interconnect them. The clusterhead selection process is also priority-based, in this case based on the remaining energy at the nodes only. Clusterheads are active during a time period based on their energy and a factor α, called the *cluster timeout factor*, that varies between 0 and 1. Once the clusterheads are chosen, the remaining nodes listen to control packets for a period of time to determine if they can connect to two or more clusterheads. If so, TECA computes a local Minimum Spanning Tree between the clusterheads to find the optimal path to interconnect them. The optimal path is found by using link costs based on the packet loss rate, the lifetime of a link, and a penalty to further minimize the number of nodes. Finally, the nodes along this optimal path become bridge nodes and the clusterhead are connected.

In [126], the authors propose a hybrid protocol based on connected dominating sets and local Delaunay triangulations. The protocol first generates a maximal independent set using a clustering technique, and then applies a localized Delaunay triangulation to create the network backbone. The authors show that the proposed algorithm produces a sparse network, i.e., with $O(n)$ links, that is also planar, and each node has a bounded degree. Further, the reduced topology is a spanner in terms of the length and number of hops. They also prove that the total communication cost of the algorithm is $O(n)$ and the computational complexity of each node is $O(d \log d)$, where d is the number of its 1-hop neighbors.

The clustering algorithm finds a maximal independent set selecting clusterheads using the smallest node id among the node's one hop neighborhood. The second step finds connectors or gateways to connect the clusterheads. This last step is performed based on two important lemmas proved by the authors. First, they prove that for every node v, the number of clusterheads with k hops is bounded by a constant $l_k \leq \frac{\pi(k+0.5)^2}{\pi(0.5)^2}$ where k is the radius of node v. Second, they also prove that, if *VirtG* is the graph connecting all pairs of clusterheads u and v, if there is a path in the unit disk graph connecting them with at most 3 hops, then *VirtG* is connected. Although building a CDS by finding gateways to connect any pair of clusterheads u and v if they are connected in *VirtG* is easy, it is not obvious how node u can find an intermediate node w in an efficient manner. The authors prove that the backbone built by the proposed method has a bounded node degree of $max(l_3, 5 + l_2) \leq 49$, a number of hops stretch factor of at most 3, and a length stretch factor of at most 6. Once the backbone is built, the authors apply the localized Delaunay triangulation technique presented in [75] to make the graph planar while preserving its spanner properties.

Another hybrid topology construction mechanism is the one presented in [132], where the authors combine clustering and adjustable transmission power. The basic

idea is to form clusters using a short transmission range and therefore reduce the node density and then allow clusterheads that are two or three hops away to use higher transmission power and connect the clusters. This longer transmission range eliminates the process of finding and selecting gateways, cluster connectors, or coordinators included in several other protocols. Once the clusters are formed and the clusterheads connected, the mechanism includes a localized connected dominating set algorithm to select a final and smaller set. This algorithm takes $O(\Delta)$ computational complexity and $O(1)$ message complexity, where Δ is the maximum node degree in the graph, when applied in sparse networks.

Part III
Topology Maintenance

Chapter 10
Introduction

10.1 Introduction

This part of the book is devoted to Topology Maintenance, the second component after Topology Construction to exercise Topology Control. This part consists of four chapters devoted to general aspects of topology maintenance, and topology maintenance static, dynamic, and hybrid techniques. This chapter defines topology maintenance and introduces and explains a new taxonomy. In addition, the chapter includes topology maintenance general design issues, triggering criteria, and mechanisms utilized to wake up sleeping nodes. At the end, the chapter includes a section that describes the assumptions and parameters that will be used in the performance evaluation of the different topology maintenance mechanisms introduced in subsequent chapters.

10.2 Definition of Topology Maintenance

Given a certain number of nodes distributed in a specific area, topology construction aims at building a reduced topology that will save energy while preserving network connectivity and area coverage. Once the initial reduced topology has been created, the network starts performing the tasks it was designed for. Every activity involved in the network's mission, like sensing, processing, and distribution of the information has an associated cost that will consume the energy of the active nodes. Therefore, in order to increase the total network lifetime, the set of active nodes, the ones in the reduced topology, cannot be active all the time. Rather, a topology maintenance mechanism should be in place to build a new reduced topology – with the collaboration of formerly inactive nodes – so that all nodes participate in the network, consume their energy in a fair manner, and increase the lifetime of the network. Accordingly,

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

> ***Topology Maintenance*** *is defined as the process that restores, rotates, or recreates the network topology from time to time when the current reduced one is no longer optimal. It involves rotating the role of nodes as much as possible to increase the network lifetime.*

Topology maintenance can be exercised in different ways depending on when the topologies are built, the scope, and the type of triggering event or metric. These options are explained next.

10.2.1 When Are the Reduced Topologies Built?

Topology maintenance techniques can be subdivided as static, dynamic, or hybrid, according to the time when the new reduced topologies are built. *Static topology maintenance techniques* calculate all different topologies during the first topology construction process. These topologies are built and stored in memory and switched out for each other when needed. As such, static techniques have "pre-planned" topologies. The best example to describe these techniques is making analogy with the lights of a Christmas tree: the entire set of lights contains a number of pre-defined subsets that cover the entire tree and take turns over time. As can be inferred, static techniques take additional time during the initial topology construction phase to calculate all additional topologies, but once this process is finished, the switching process is faster than if a new topology construction phase had to take place. Further, the communication overhead of each subsequent topology construction phase is also saved. Static topology maintenance techniques may have some drawbacks too. For example, it is difficult to know a priori how each of the topologies and their nodes will consume their energy. Therefore, the mechanism may choose to use some nodes in more than one topology that may not be available or will make the topology to last shorter than expected.

Dynamic topology maintenance techniques, on the other hand, calculate a new reduced topology "on the fly", triggering the topology construction mechanism when necessary. Dynamic topology maintenance mechanisms have the advantage of having more information about the network to find a more appropriate new reduced topology. The disadvantage of these mechanisms is that they consume additional resources every time they are run, therefore, it is extremely important that the topology maintenance and the topology construction mechanisms are both very energy efficient.

Finally, *hybrid topology maintenance techniques* use both, static and dynamic techniques. Hybrid techniques calculate all different reduced topologies during the first topology construction phase (static approach) but if the coming topology cannot be established because the sink has no connectivity with the nodes (dead topology), the mechanism finds a new topology on the fly (dynamic approach). This approach inherits some of the advantages and disadvantages of both the static and dynamic techniques.

10.2.2 Scope of Topology Maintenance

The second aspect is related to the scope of the technique: which nodes, or which part of the network, are involved in the execution of the topology maintenance al-

gorithm. The scope can be global or local. While *global techniques* consider all the nodes in the network in order to make a global-optimal decision, *local techniques* only consider a small subset of the nodes in order to make a decision in a local-optimal fashion. Therefore, global techniques switch the entire topology, and local techniques only switch a portion of the network, such as a branch of a tree, a cluster, or even just one node.

Although a local strategy could be used in static techniques, this option is rarely utilized given its high complexity. For example, a static local technique should be able to pre-calculate a very high number of different options on a per-node basis. Also, cluster- or tree branch-based schemes might need nodes from other nearby clusters or branches, which would trigger a chain reaction that might make the local strategy look like a global one, but with more overhead.

10.2.3 Triggering Criteria

Whether the topology maintenance mechanism is static, dynamic, or hybrid, global or local, there is one important question related to all: what is the criterion or criteria that will be used to trigger the process of changing the current topology? The triggering criteria, which may have important implications in terms of energy savings as well as coverage, reliability, and other important metrics, may be based on one of the following choices:

- **Time-based:** In time-based topology maintenance, the current topology is changed every time a timer expires. The amount of time is usually fixed and pre-defined, and is a very critical variable. Too short a time between changes may cause unwanted, extra overhead as a consequence of switching the topology more often than necessary. On the other hand, a very long time between changes may use a suboptimal network longer than necessary, with the possibility of losing important events due to its poor coverage.
- **Energy-based:** Given the energy limitations of wireless sensor devices, it makes sense to switch the topology when the energy level of the nodes goes below a certain threshold. Again, the choice of the energy threshold is very critical for the same reasons explained before. Changing the topology too often may end up spending more energy than the energy that is supposed to be saved by topology maintenance, thereby defeating its purpose. A very low threshold, on the other hand, will make certain critical nodes unavailable for upcoming topologies.
- **Random:** In random-based topology maintenance, the current topology is switched using a random variable.
- **Failure-based:** A failure-based technique triggers the process of changing the current topology after a node, or a number of nodes, has failed. These techniques require the existence of failure detection and notification mechanisms.
- **Density-based:** Another criterion might be based on the density of the network. A similar metric might be the node degree of the network or the node degree of some important nodes.

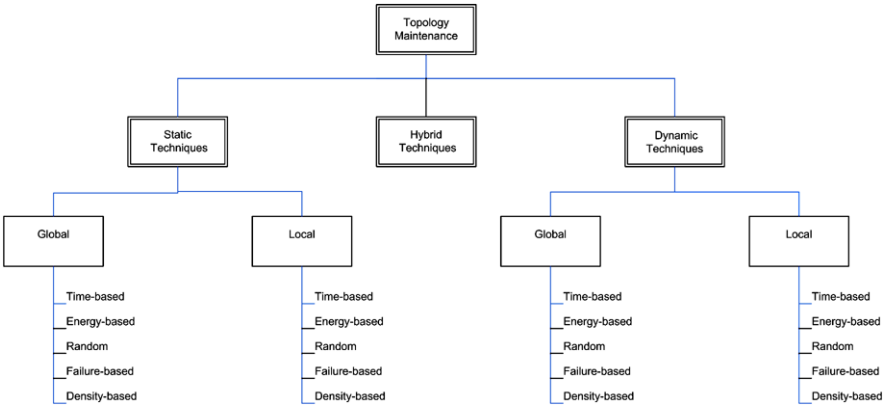


Fig. 10.1 Classification of topology maintenance.

- **Combinations:** Combinations of these criteria could be used as well. For example, the topology maintenance could be activated based on energy and failure, or time and energy.

Figure 10.1 shows a complete taxonomy of topology maintenance that also includes some of the possible triggering criteria.

The following three chapters describe in more detail some of the most important techniques in each category. Performance evaluations of the most important techniques are also included to provide additional insights about the usefulness of each technique. Chapter 11 is devoted to static techniques, Chapter 12 covers dynamic techniques, and Chapter 13 includes hybrid techniques as well as a global comparison.

The rest of this chapter is devoted to aspects that apply to all topology maintenance techniques to be covered in the next chapters, such as topology maintenance general design issues, radio synchronization, and the description of the common simulation scenarios and parameters that will be used in the performance evaluations included in the following chapters.

10.3 Design Issues

Topology maintenance, as part of the entire topology control process, shares most, if not all, of the design criteria established for topology construction. In order for this process to be effective and provide the expected results in terms of extending the lifetime of the network, topology maintenance mechanisms must be designed with careful consideration to the following aspects:

- **Energy efficiency:** As with any other function in wireless sensor networks, the topology maintenance technique must be energy efficient too. The energy savings derived from switching the current topology should not be used by the topology maintenance process, as there will not be any overall energy benefit. This is

not to be confused with the energy efficiency of the topology construction technique. For example, a dynamic global topology maintenance technique may be very energy efficient but if the topology construction technique that is called upon to change the topology is not, then the network might be better off not doing topology maintenance at all.

- **Low overhead:** The topology maintenance technique should not inject a considerable amount of control packets. Some techniques, such as those based on failures and energy need to implement a notification mechanism so the sink node can initiate the process.
- **Low complexity:** The algorithms used in topology maintenance techniques must be simple.
- **Even energy distribution:** Topology maintenance techniques somehow should try to distribute the energy consumption in an even manner among all the nodes in the network. The topologies should be changed so that all nodes have a similar participation in the network.
- **Low convergence time:** During the topology maintenance process a current topology will be replaced by a new one, therefore there will be a transition time during which the network might not be active. This time must be as small as possible. Static techniques offer a clear advantage in this aspect, as the new topology has already been calculated. In dynamic techniques, this time will be longer and depends on the convergence time of the topology construction mechanism.
- **Memory consumption:** Wireless sensor devices don't have a lot of memory. Some topology maintenance techniques may need a considerable amount of memory, such as those static global techniques that need to store all pre-calculated topologies.

10.4 Synchronizing Radios

Another common issue in all these mechanisms is how to wake up sleeping nodes so that they can participate in a new topology construction procedure. For example, at the time of network deployment, all nodes are active. Then, the first topology construction process takes place, and a new reduced topology starts to work. Now, whenever the topology maintenance procedure calls for the total or partial replacement of the reduced topology, the nodes that were sleeping before need to be woken, so they can participate in the new topology construction process and contribute to extend the network lifetime. This synchronization is not simple, and there are several ways to achieve it.

One way to wake up all the interested nodes at the appropriate time is to have a completely synchronized network. For time-based topology maintenance techniques, this global synchronization is especially useful. However, network wide synchronization is not easily achieved, especially in large networks.

In [58], the authors present two wake up strategies to deal with the synchronization problem in dense networks. The first one is completely random and is based on the previous knowledge of the minimum number of nodes required to be woken up

to be able to build a reduced connected topology that also covers the area of interest. This last aspect is further explained in Section 11.3.

Under the **random scheme**, the nodes to be woken up are uniformly chosen at random from the area of interest. This is similar to each node waking up by itself in an independent manner with a fixed probability. The good thing about this scheme is that it requires no knowledge about the network topology and location of the nodes, and can be implemented with no overhead. Similar to the results obtained in Chapter 7 related to the giant component and connected topologies, the authors found that with a radius of $r = 10$ the scheme produces a connected topology that covers 90% of the area with a network density of 0.0095. However, in order to provide 100% coverage, the network density had to be increased considerably, to 0.014.

The second technique is called **patterned-wakeup**, which can be implemented in a fully distributed manner in dense networks with no overhead if each sensor knows its location. The scheme is called patterned because nodes are woken following by a specific pattern. For example, the authors study the establishment of the reduced topology in a square grid of size D in which the nodes that make up the topology are those located in the intersection of the internal grid lines. Two additional patterns included in the work are based on an hexagonal grid and a strip. One drawback of these schemes is that they all assume that the network is so dense that they can find a node to wake up in any location. Since this assumption might not be true in practice, in [58] some approximations are included to deal with this issue.

Nodes can also be woken using a **metric-based scheme**. This technique is based on the local calculation of a metric of interest, such as the number of redundant nodes and the density of active nodes. The protocols GAF [137], CEC [135], and SPAN [20] use this approach to maintain the set of active nodes in the network. The main difference between the protocols GAF and CEC is that GAF uses location information to determine redundancy, while CEC and SPAN are neighbor-based protocols that determine redundancy based on neighbor lists. Every sleeping node in the network will wake up in a periodic way in order to check its own status as a redundant node. If it is not needed, it will go to sleep again for a period of time. On the other hand, if it detects that it is not a redundant node anymore, it will remain in active mode.

One **density-based** technique proposed in [136] determines the sleeping time of the nodes proportionally to the number of active nodes that a node can listen to during its active time. This technique is meant to reduce the amount of active nodes if enough nodes are awaken, and increase them otherwise. A similar technique, proposed in [42], works with the theory of percolation or node self-organization. All nodes start in sleeping mode and wait a random amount of time. Once active, the nodes poll their neighborhoods to determine the number of active nodes and start a timer. The nodes go to sleep again once the timer expires or when they detect that the number of active nodes that responded to the poll is above the minimum expected. The main idea is to reach a synchronized state of oscillation in which different sets of nodes will be active at the same time and will keep the desired minimum active

node density. In [42], the authors show that a value of 6 active neighbors provide an almost completely connected reduced graph.

Another option is that of nodes being equipped with **two radios**, one for the transmission of data and a cheaper one for control messages only. While the main radio is put to sleep as much time as possible, the secondary radio, which is cheaper in terms of cost and energy consumption, is kept active at all times to listen to control messages, such as those used to perform topology maintenance. This is the approach presented in [45] where a separate super low power radio is used to wake the main radio up. From the energy point of view, this approach makes a lot of sense, since wireless sensor networks are low duty cycle systems where nodes only spend a very small part of their time transmitting real data. From the cost point of view, this approach increases the cost of the wireless sensor devices. A similar scheme is proposed in [108] but the authors don't limit themselves to having one ultra low power radio for control data and one radio for data transmissions. Instead, they trade energy savings versus latency by equipping the nodes with two radios, using the most efficient one available for data transmission, and putting the other one into low power listen mode.

10.5 Performance Evaluation

This section includes common information about the simulation-based performance evaluation that will be carried out in the next three chapters to assess the performance of the various static, dynamic, and hybrid topology maintenance techniques. The purpose of the experiments is to determine the benefit of implementing topology maintenance techniques in both sparse and dense networks, and compare their performance versus the choice of not implementing topology maintenance at all. In the performance evaluation, the following common assumptions are made:

- All nodes are located in a two dimensional space and have a perfect communication coverage disk.
- Nodes have no information about their position, orientation, or neighbors.
- The initial graph, the one formed right after the deployment, is connected.
- Distances can be calculated as a metric perfectly proportional to the Received Signal Strength Indicator (RSSI).
- There is no packet loss at the Data Link Layer.
- There is a way in which a node can be awakened when its radio is off.

In these experiments, sparse topologies are defined as topologies in which the communication radius is calculated based on the Critical Transmission Range (CTR) formula of Penrose–Santi [105] (Equation 7.1 described in Section 7.2). This guarantees that the node degree is very low, creating a weakly-connected topology. Dense topologies are defined as sparse topologies in which the original number of nodes is doubled. Each point in the graphs is the average of running the same experiment 150 times, i.e., with 150 different topologies.

Table 10.1 Simulation parameters.

	Sparse Topologies	**Dense Topologies**
Deployment area	200 m × 200 m	
Number of nodes	50	100 and 400
Number of sinks	1 sink	
Number of topologies (runs) per experiment	150	
Transmission Range	1 × CTR(50) equivalent to: 37 m [105]	
Node Distribution	Uniform (200, 200)	
Time Threshold	1000 time units	
Energy Threshold	10% of total energy	
Max number of reduced topologies (static and hybrid schemes)	3 reduced topologies	
E_{max}	1 Joule	
A3 Weights	$W_E = 0.5$, $W_D = 0.5$	
Energy Consumption	Eelec = 50 nJ/bit; Eamp = 10 pJ/bit/m^2	
	Short Messages = 25 Bytes	
	Long Messages = 100 Bytes	
	Idle state energy consumption assumed negligible	

In all simulations the A3, EECDS, and CDS-Rule-K topology construction algorithms described in Chapter 8 were utilized. The implementations were coded and tested in the Atarraya simulation tool described in Appendix A, which is designed with the purpose of testing topology control algorithms. In all simulations, the wireless sensor devices are uniformly distributed in an area of interest of 200 m × 200 m; the number of nodes is varied to create sparse and dense networks; and all scenarios have one sink. Each active node in the current reduced topology is scheduled to send data messages directed to the sink every 10 time units. Since all topology construction algorithms produce a tree-based reduced topology rooted at the sink, a very simple routing algorithm is implemented in which nodes forward the data messages to their respective parents. In all experiments, no data aggregation or similar strategy is implemented. Energy is drained when a packet is either sent or received according to the energy dissipation model presented in Section 2.3. Table 10.1 includes a summary of the most important simulation parameters used in all simulations.

Chapter 11
Topology Maintenance Static Techniques

11.1 Introduction

This chapter presents static topology maintenance techniques, which, as explained in Chapter 10, calculate all different topologies a priori, during the initial topology construction time, and switch them once the triggering criteria is met, whichever it might be.

Static techniques are usually global in nature, as the calculation of replacements for more localized portions of the network may take an excessive amount of resources. One alternative to make static local techniques more attractive is to increase the scope of the portion of the network to switch. For instance, in a tree-based network, several different tree branches might be found during the first topology construction phase and, upon the triggering criteria, an entire branch is switched. However, as the scope is increased, local techniques become more and more similar to global techniques. In this chapter we only analyze the performance of global techniques.

One important decision to make in the construction of the pre-planned topologies is whether or not to build completely node-disjoint topologies. This aspect must be seen along with the density of the network. For example, in a very sparse network, it might be very difficult to find more than one completely disjoint topology, and in that case, the topology maintenance mechanism would be switching the same topology over and over again. The opposite is true for highly dense networks, in which many node-disjoint reduced topologies might be found. Nonetheless, regardless of the network density, in most situations, it might be a good idea to allow shared-disjoint topologies. First, there might be critical nodes that, if not included in the topologies, would make static techniques look like topology construction algorithms. Second, shared-disjoint topologies allow new nodes to be included in the network with the possibility of extending its lifetime.

In either case, a series of pre-planned topologies are stored in the sink node and switched based on the established triggering criteria. This static global technique is called the **Christmas tree** technique, a rotation technique that will behave similar

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

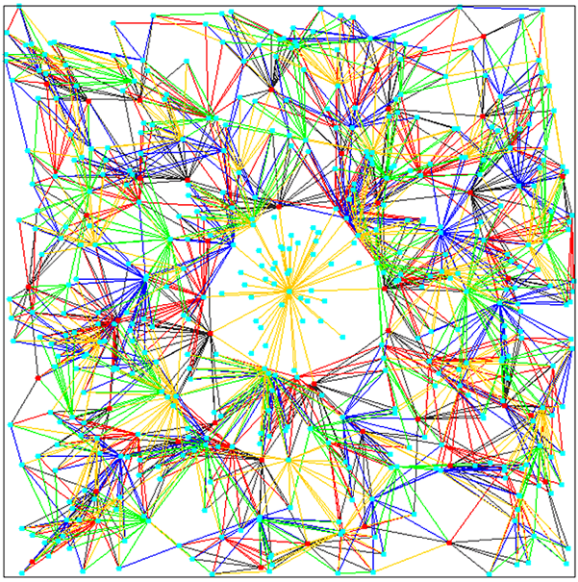


Fig. 11.1 Multiple node-disjoint CDS trees over the same network.

to the lights in a Christmas tree, where the sets, in this case completely disjoint, take turns to light up (cover) the entire tree (area). Figure 11.1 shows an example of the number of node-disjoint reduced topologies that can be created to implement the Christmas tree approach to static global topology maintenance.

11.2 Performance Evaluation of Static Global Topology Maintenance Techniques

This section presents a performance evaluation of the static global Christmas tree topology maintenance technique carried out in the simulation tool Atarraya (Appendix A) using the simulation parameters and assumptions described in Chapter 10. The purpose of the experiments in this section is to determine the benefit of implementing static global topology maintenance techniques in both sparse and dense networks, and compare their performance versus the choice of not implementing topology maintenance at all. The following topology maintenance techniques and triggering criteria are included:

- **No Topology Maintenance – No TM:** The initial reduced topology works permanently until the sink detects that it does not have any more active nodes in range. In general, this is the same termination policy for all the algorithms.
- **Static Global Time-based Topology Rotation – SGTTRot:** Every pre-determined time interval the topology maintenance algorithm rotates the active reduced topology for one of the pre-planned ones.

- **Static Global Energy-based Topology Rotation – SGETRot:** Every time a node reaches a critical energy threshold, the topology maintenance algorithm rotates the active reduced topology for one of the pre-planned ones.

In order to implement these two static global techniques, two types of messages are implemented in the algorithms: the *Notification Message* and the *Reset Message*. The notification message is utilized by the nodes to notify the sink when they are running out of energy, so the sink knows that it is time to rotate the topology. The reset message, on the other hand, is used by the time-based and energy-based techniques. Once the sink knows that it has to switch the topology, it broadcasts the reset message with the ID of the new tree, so that the new pre-planned topology becomes active.

For simplicity, in the experiments, the number of pre-planned topologies is limited to three.[1] The process of selecting these three topologies is different depending on the underlying topology construction mechanism. In the case of A3, its selection metric is manipulated to lower the probability of selecting one node in more than one topology. However, since the same topology construction algorithm is run every time, if a particular node that has been used before in another topology is needed to guarantee network connectivity, it may be selected again. In other words, the static global technique based on A3 may produce shared-disjoint topologies.

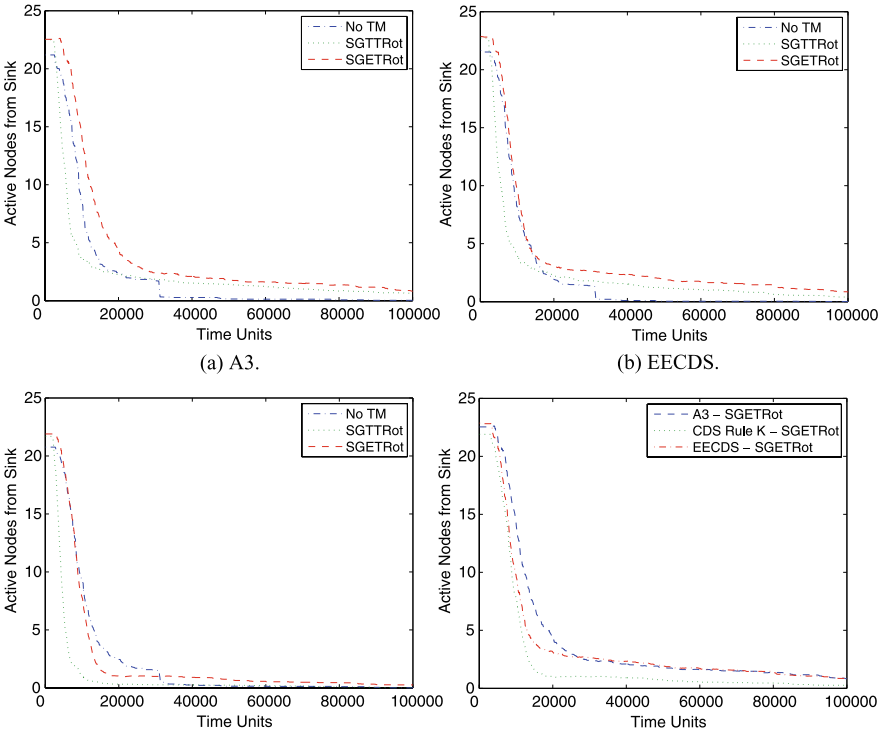
In the case of the EECDS topology construction algorithm, it also uses a numerical metric to perform the selection of the active nodes, which allows the application of a penalty in this metric to all nodes that have been selected to be active in other subsets. However, depending on the density of the network, the metric cannot be changed in all the nodes. EECDS works in two phases, selecting Black nodes in the first phase, and Blue and Gray nodes in the second phase. Black nodes are backbone nodes that act as clusterheads, Blue and Gray nodes are then used to interconnect the clusterheads. As a result of this two-phase approach, EECDS can only reduce the selection metric to the Black nodes that have been selected previously to be active in other subsets. In dense topologies it is expected that if there is a change in the selection metric of the Black nodes, the set of elected Gray nodes will also change. However, in sparse topologies the case is different because there are so few possibilities of selection of Gray nodes that they will tend to be the same ones in every subset.

Finally, in the case of the CDS-Rule-K topology construction algorithm, since it does not include any numerical metric to select the nodes, the algorithm is left unchanged, meaning that it may produce very similar trees.

11.2.1 Sparse Networks

Figure 11.2 shows network lifetime simulation results when performing static global topology maintenance in sparse networks. Two common observations can be made. First, static global topology maintenance in sparse networks can either improve or

[1] A more complex mechanism that finds as many as possible completely disjoint topologies can be found in [58].



(a) A3. (b) EECDS.

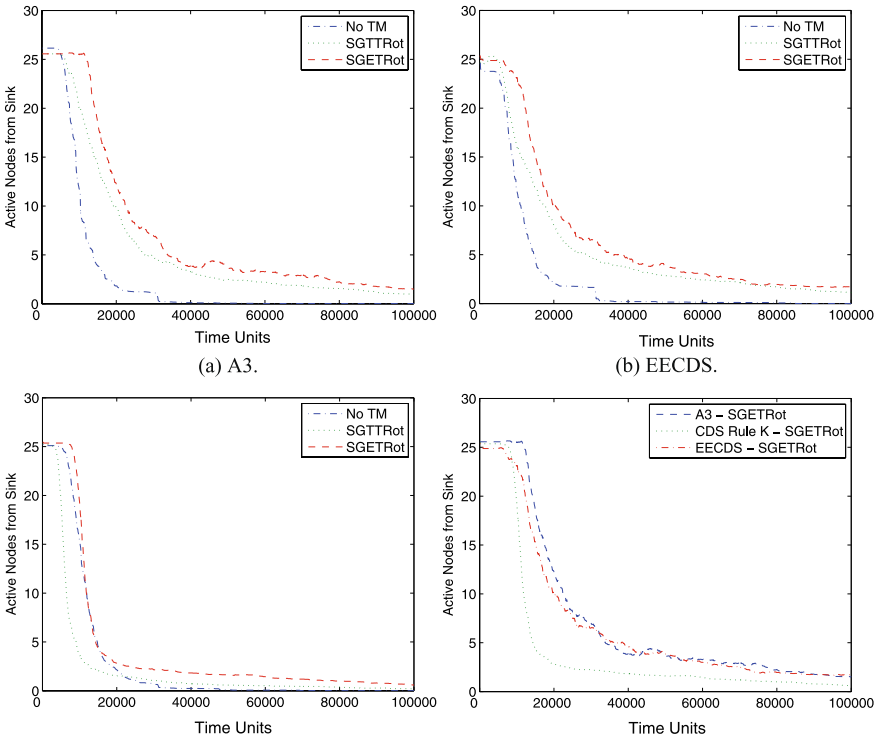
(c) CDS-Rule-K. (d) Comparison of best mechanisms.

Fig. 11.2 Network lifetime with and without static global topology maintenance using the A3, EECDS, and CDS-Rule-K topology construction mechanisms in sparse networks.

degrade the network lifetime compared with the option of not performing topology maintenance at all. Second, the performance is not significantly better or worse, meaning that applying static global topology maintenance techniques in sparse networks might not be useful.

These results are totally expected, as in sparse networks not many disjoint topologies can be created. Remember that in this scenario only 50 nodes are spread in an area of 200 square meters. This means that the topology maintenance procedure just activates the same topology, or a very similar reduced topology, every time and the overhead related to the changing process drains extra energy, killing the nodes earlier. The impact of not having disjoint subsets can be appreciated in the CDS-Rule-K algorithm (Figure 11.2(c)), as both static techniques did not even reach the performance of having no topology maintenance at all.

Another important observation is that the performance of the static global techniques may change according to the value of the triggering mechanism, i.e., the value of the timer and the energy threshold. Nonetheless, it is not expected that changing these values to more appropriate (optimal) ones will produce considerably better or worse results, as the effect of the reduced number of disjoint topologies



(a) A3. (b) EECDS.

(c) CDS-Rule-K. (d) Comparison of best mechanisms.

Fig. 11.3 Network lifetime with and without static global topology maintenance using the A3, EECDS, and CDS-Rule-K topology construction mechanisms in dense networks.

will prevail over the threshold values. A sensitivity analysis looking at the effect of these values in the performance of the techniques is included in Section 13.4 for dense networks.

Finally, Figure 11.2(d) compares the best performing techniques for each algorithm. As it can be observed, all energy-based techniques perform better than the time-based techniques. This result is expected as the energy threshold does not make the topology to change until the nodes are almost without energy. Since there are not many disjoint topologies, the energy-based techniques do not waste as much energy as the time-based techniques in the switching process, which switch the topology every 1000 time units. Of the three techniques, the one based on the A3 algorithm is the best performing. This is because A3 is more energy efficient than EECDS and CDS-Rule-K, as it has been demonstrated in [128].

11.2.2 Dense Networks

The performance of the static global topology maintenance techniques is expected to improve with the network density, as it increases the chance of having more

node-disjoint topologies. This improvement in performance is, in fact, shown in Figure 11.3, which includes the results for more dense networks, those using 100 nodes instead of 50. Comparing Figures 11.2 and 11.3, it can be observed that while in sparse networks the no topology maintenance option was either the best performing one, or very close to be the best performing one, in dense networks the opposite is true. Actually, it can be seen from the figures that all topology maintenance techniques under consideration extend the network lifetime over the no topology maintenance option, so their use is more than justified.

Two additional observations are worth including. First, the performance of the topology maintenance techniques depends on the underlying topology construction algorithm. It can be seen that while the performance of the A3-based and EECDS-based techniques improved with respect to the same experiments in sparse networks, the performance improvement of the CDS-Rule-K-based technique is not as big. This has to do with the way of selecting the disjoint trees, as explained at the beginning of this section. A3 and EECDS allow the manipulation of the node selection metric, and, therefore, they are able to build more node-disjoint trees. Second, as in the case of sparse networks, the techniques that used the energy triggering metric outperformed the time-based techniques. Again, this has to do with choosing an optimal switching time or energy threshold, which has not been attempted thus far. A sensitivity analysis is included in Section 13.4, which also looks at the effect of increasing the network density even further.

11.3 Other Static Techniques

There are not many other topology maintenance techniques that could be classified as static global. One interesting approach based on the Christmas tree concept is described in [58]. In their work, the authors introduce algorithms to find the maximum number of node-disjoint reduced topologies in a dense network. The schemes are designed so that the reduced topologies are not only connected but also cover the entire region of interest, or at least, a big portion of it. The schemes use the random- and pattern-based approaches described in Section 10.4 to wake up the nodes and build the connected-covered reduced topologies.

Assuming a two-dimensional plane, a single sink, perfect circular area of coverage per node r, and communication coverage equal to sensing coverage, the authors prove that the optimal network density in number of nodes per unit area required to provide a reduced connected topology that covers the area of interest is given by:

$$d_{OPT} \geq \frac{0.522}{r^2} \tag{11.1}$$

In other words, Equation 11.1 above states that no reduced topology can guarantee connectivity and coverage if the network density is lower than $(0.522/r^2)$. The authors show that the random scheme and the patterned square-grid, hexagonal-grid, and the strip-based techniques, require a network density given by:

$$
\begin{aligned}
d_{Random} &= (3\text{–}4 \text{ times}) \times d_{OPT} \\
d_{Square} &= \frac{1}{r^2} \\
d_{Hexagonal} &= \frac{0.769}{r^2} \\
d_{Strip} &= \frac{0.536}{r^2}
\end{aligned} \tag{11.2}
$$

Once the initial reduced connected-coverage topology is built using any of the mechanisms mentioned above, the chosen mechanism is run iteratively using only the remaining nodes, or those nodes that have not been part of any reduced topology calculated before. Since the work focuses on building the maximum number of node-disjoint topologies, it does not include any performance evaluation of this Christmas tree-based technique, nor does it mention anything about the triggering criteria that could be utilized to switch the topologies.

Chapter 12
Topology Maintenance Dynamic Techniques

12.1 Introduction

Contrary to static topology maintenance techniques, dynamic techniques do not make a priori calculations to determine the topology that will become active when the current one is no longer appropriate. Instead, dynamic techniques make those calculations "on-the-fly". As such, dynamic techniques are usually more time and energy consuming since they have to run the topology construction process several times. Therefore, the energy efficiency of the underlying topology construction algorithm plays an important role. However, since dynamic techniques consider the current status of the network when making the new calculations, the topology construction process usually chooses an optimal or close to optimal topology every time it is run, resulting in better or more adequate subsequent topologies, as compared with static techniques.

Dynamic topology maintenance techniques can also be subdivided into global or local according to the part of the topology that they switch. Global techniques run a network wide new topology construction process every time the topology needs to be changed and, therefore, switch the entire topology. Local techniques, on the other hand, run a more localized process meant to calculate and establish a new branch of a tree, a cluster, or a specific link or small set of links. Contrary to static techniques, local dynamic techniques might be able to increase the network lifetime more than either static or dynamic global techniques, as it may be possible to restore a local failure with just a few computations and a small number of messages. As in the case of static techniques, dynamic topology maintenance techniques may be activated by the same triggering criteria defined in Chapter 10.

In this chapter, dynamic global and local topology maintenance techniques are described and evaluated in sparse and dense networks.

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

12.2 Performance Evaluation of Dynamic Global Topology Maintenance Techniques

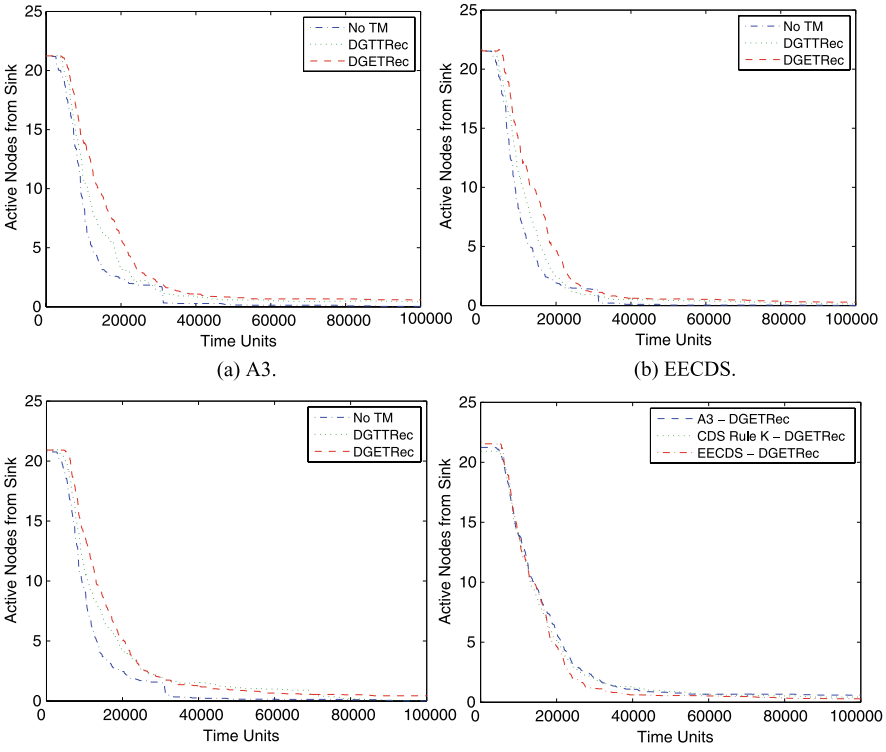
Dynamic global topology maintenance techniques change the entire topology by running the topology construction mechanism anew every time it is needed. Using the same network parameters described in Table 10.1, simulation experiments were carried out to assess the network lifetime with dynamic global topology maintenance techniques in sparse and dense networks using the A3, EECDS, and CDS-Rule-K topology construction algorithms compared against the case where no topology maintenance is performed. The following topology maintenance techniques and triggering criteria are included:

- **No Topology Maintenance – No TM:** The initial reduced topology works permanently until the sink detects that it does not have any more active nodes in range. This is the same termination policy for the rest of the algorithms.
- **Dynamic Global Time-based Topology Recreation – DGTTRec:** Every set time interval the topology maintenance algorithm terminates the previous reduced topology and invokes the topology construction algorithm to create a new one.
- **Dynamic Global Energy-based Topology Recreation – DGETRec:** Every time a node reaches a critical energy threshold, the topology maintenance algorithm terminates the previous reduced topology and invokes the topology construction algorithm to create a new one.

12.2.1 Sparse Networks

Figure 12.1 shows the network lifetime simulation results using dynamic global topology maintenance techniques in sparse networks using energy and time triggering criteria. Contrary to the static global technique results for sparse networks shown in Figure 11.2, now all topology maintenance techniques provide better performance than the case with no topology maintenance. It is clear that the new topology construction process is able to find better topologies than the pre-planned algorithms, extending the network lifetime, even in sparse networks. However, the performance improvement over the no topology maintenance option is not considerable either, reinforcing the conclusion reached before that, in sparse networks, topology maintenance does not provide important benefits. Comparing Figures 12.1 and 11.2 again, it can be observed that dynamic techniques provide better performance than static ones, which is expected given that more and better topologies can be built using dynamic methods. A more detailed comparison among these techniques is included in Section 13.3.

As in the case of static global techniques, from Figure 12.1(a), the same two conclusions can be drawn. First, all energy-based techniques outperformed the time-based techniques. Again, these performance results might vary depending on the time period and the energy threshold chosen. Similarly, in the case of A3, the behavior can also be changed varying the weights in Equation 8.2, which would give



(a) A3. (b) EECDS.

(c) CDS-Rule-K. (d) Comparison of best mechanisms.

Fig. 12.1 Network lifetime with and without dynamic global topology maintenance using the A3, EECDS, and CDS-Rule-K topology construction mechanisms in sparse networks.

preference to those nodes that contain more energy, or those nodes that are closer to the parent node. There are clear trade offs using these variables and weights, which are beyond the scope of the book at this time. Second, the technique using the A3 topology construction algorithm is the best performing one, although not by an important margin. This is due to A3's lower and linear message complexity compared with EECDS and CDS-Rule-K [128].

12.2.2 Dense Networks

More interesting results are obtained in the case of dense networks, as can be observed from Figure 12.2. Now the figure shows that by increasing the number of nodes from 50 to 100, dynamic global topology maintenance techniques are able to keep active a considerably higher number of nodes compared with the option of no topology maintenance. Compared with the case of dynamic global techniques in sparse networks shown in Figure 12.1, the performance in dense networks is nearly twice as good as the performance in sparse networks, meaning that more and better trees can be found in dense networks.

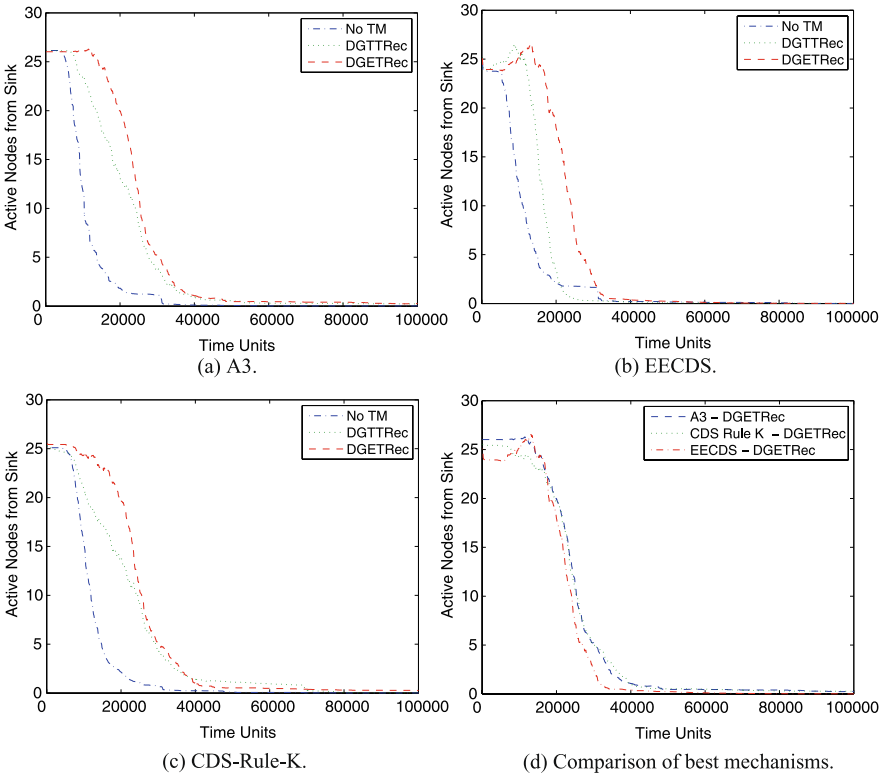


Fig. 12.2 Network lifetime with and without dynamic global topology maintenance using the A3, EECDS, and CDS-Rule-K topology construction mechanisms in dense networks.

Also, as in the case of static global techniques in dense networks, here the performance provided by the energy-based techniques outperformed the ones based on time. As stated before, this does not mean that time-based techniques are better or worse than energy-based techniques; it just means that they provided a worse performance with those particular settings, i.e., 1000 time units for the time-based criterion, and 10% of the total remaining energy for the energy-based criterion. More insights about this aspect are included in Section 13.4.

Figure 12.2(d) plots the best performing mechanisms in each case. As in previous cases, it seems that the topology maintenance mechanism based on the A3 topology construction algorithm has a better performance than the ones based on the CDS-Rule-K and EECDS algorithms, however, the advantage is marginal. Although A3 has better message complexity than CDS-Rule-K and EECDS, the small amount of sensors (100) and the small energy threshold (10%) cause the topology maintenance algorithm to call the topology construction algorithm very few times, which makes the amount of energy spent by the three schemes very similar. A more pronounced lifetime advantage of A3 would had been seen if the topology construction algorithm had been run many more time.

12.2.3 Other Dynamic Global Techniques

There are topology construction mechanisms that embed and utilize a dynamic global topology maintenance technique without formally calling it topology maintenance. One example is that of the LEACH protocol described in Sections 3.2.2, 4.2.2, and 8.3. LEACH changes the entire network topology periodically, in *rounds*; therefore, it is a dynamic global technique with a time-based triggering criterion. As described in Section 8.3, it also includes a clusterhead selection metric that guarantees that the nodes consume their energy in an even and fair manner.

12.3 Performance Evaluation of Dynamic Local Topology Maintenance Techniques

In dynamic local topology maintenance, once the initial topology is built using any of the available topology construction algorithms, only a portion of the network works toward restoring or recreating part of the topology, in one of three ways: by changing enough nodes to change an entire branch of a tree, by changing a few nodes within the branch, or by changing an entire cluster. The idea is, given that the energy level of an active node has reached its energy threshold and needs to be replaced, instead of replacing the entire network topology, a localized mechanism could replace this node with much less overhead, therefore extending even further the network lifetime.

Although in this case the triggering condition is energy-based, a node may also ask for its replacement in a random manner, based on its failure, based on the node density around it, or any other criteria. Further, it is possible that the failure of one node does not cause a considerable loss to the network therefore, it might be better to wait until more nodes cease to operate in order to start the recovery procedure. As such, metrics can be defined to evaluate the impact of the losses and decide whether to trigger the replacement procedure or not. Examples of these metrics are presented in [86], where the authors define the following metrics:

- Cumulative reduction of sensing coverage.
- Amount of energy increase per node.
- Local reduction of sensing coverage.

The *cumulative reduction of sensing coverage* metric evaluates the total area that is not covered anymore by the failed node and all the other nodes that depended on it. One important aspect is the determination of whether the failed node is an articulation point, i.e., a node that if removed would disconnect the network. The *amount of energy increase per node* metric is related to the amount of energy that will be required to send packets to the sink node. If the failed node is on the shortest path of a set of nodes, the total cost to transmit the packets will increase. The *local reduction of sensing coverage* metric is a local version of the first one: it accounts for the amount of non-overlapping area that the failed node leaves without coverage.

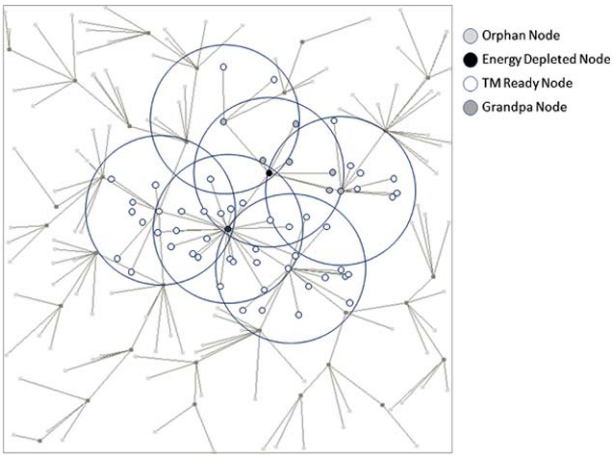


Fig. 12.3 Phase one of the DSR-based dynamic local topology maintenance technique.

It is worth mentioning that topology maintenance local techniques have rarely appeared in the wireless sensor network literature thus far, and more research is needed. Nonetheless, there is a complete body of related literature on network survivability from which proven ideas can be borrowed. Network survivability techniques have been designed and successfully applied in telephone networks, SONET-based optical networks, wireless cellular networks, and more recently in IP/WDM optical networks with protection and restoration mechanisms at the path and link level. The interested reader is encouraged to search and review the literature on network survivability.

As an example, an energy-based dynamic local topology maintenance technique is presented and analyzed in this section, which is based on the well-known Dynamic Source Routing (DSR) [16] protocol for wireless ad hoc networks. This local technique works in two phases. During the first phase, the affected node, N_i, i.e., the node with its energy level below the threshold, broadcast a wake up message that reaches all its covered and sleeping children, which now become semi-active. The message is also heard by N_i's active parent P_i and child C_i, which were part of the reduced tree. Upon receiving this message, both, P_i and C_i broadcast the message again among their respective children to wake them up too. Further, the parents of P_i do the same, so at the end of this broadcast period, all nodes two hops away on N_i's children side, and three hops away on its parent side, are semi-active. This is done in this way to guarantee that with algorithms like A3, which builds trees with very spread out branches, the procedure will always find alternate nodes to restore the topology while covering all the nodes that were previously covered by node N_i. Figure 12.3 shows an example of the first phase of the DSR-based dynamic local topology maintenance technique just described.

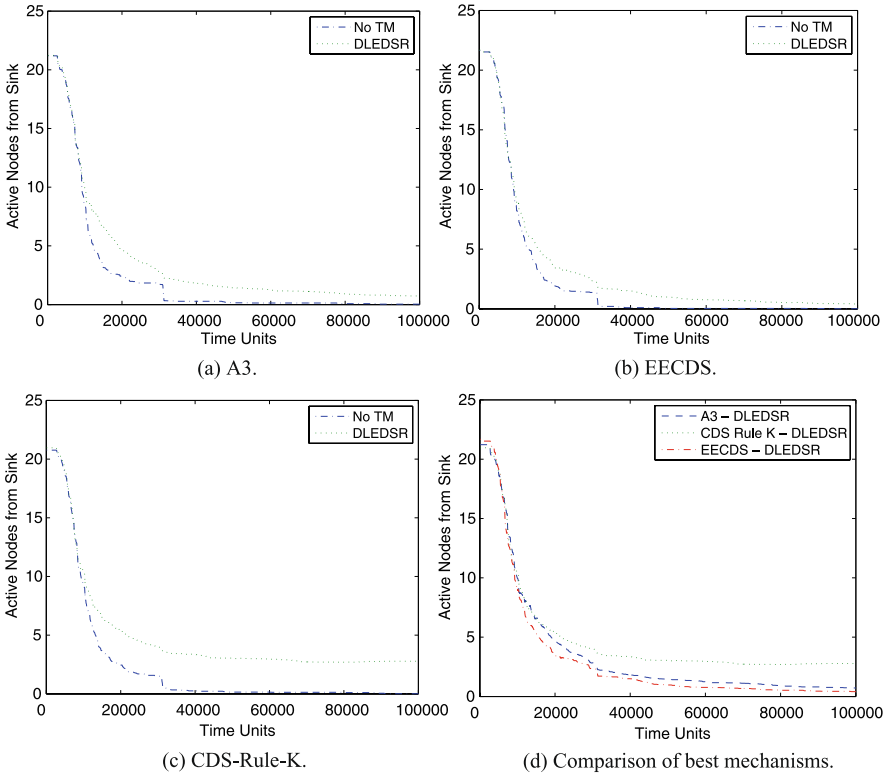
The second phase of the protocol begins with the parent node P_i doing a restricted flooding of a *Route Request Message*. The flood is restricted because it is sent to all semi-active nodes only. This route request message includes the list of node N_i's children that are now uncovered as well as space for the nodes to append routing and energy-distance information. Therefore, upon receiving a route request message, a node appends its route information and its new energy-distance metric, and sends it again to its neighbors. These messages will eventually reach node N_i's uncovered children, which may actually receive more than one message. Once all route request messages have been received, the uncovered nodes (N_i's children) select the best route back to the parent node P_i based on the route and metric information contained in the messages. At that time, they unicast a *Route Reply Message* back to the parent P_i using the chosen best route. Upon receiving this route reply message, a semi-active node changes its status to active and forwards the message to the following node in the route list. This process continues until P_i receives all route reply messages from all N_i's uncovered children. At the end of this process, if an active node that was previously inactive does not receive any route reply message, it turns itself off. The whole process guarantees that all uncovered nodes are now covered again and the tree branch is connected again.

This dynamic local topology maintenance technique was also implemented in the Atarraya simulator and evaluated using the same parameters and scenarios utilized before. In the next subsections the results in sparse and dense networks are presented where the following topology maintenance techniques and triggering criteria are included:

- **No Topology Maintenance – No TM:** The initial reduced topology works permanently until the sink detects that it does not have any more active nodes in range. This is the same termination policy for the rest of the algorithms.
- **Dynamic Local Energy- and DSR-based Repair – DLEDSR:** Every time a node reaches a critical energy threshold, the local topology maintenance algorithm based on the DSR protocol repairs the topology.

12.3.1 Sparse Networks

Figure 12.4 shows the simulation results of the dynamic local topology maintenance technique compared with the case of no topology maintenance in sparse networks using the A3, EECDS, and CDS-Rule-K topology construction algorithms. As it is clearly shown, the dynamic local technique enhances the network lifetime in all cases, meaning that, even in sparse networks, it is worth applying. Another observation is that in this case, the technique based on the CDS-Rule-K topology construction mechanism is the best performing one (see Figure 12.4(d)). This is because the CDS-Rule-K algorithm somehow builds trees with close branches, and the local procedure is able to restore the connectivity of the branch with very few nodes, as compared with A3 and EECDS. This effect is better seen after the topology has been run for some time, after 15000 time units in the figures. At that time, many



(a) A3. (b) EECDS.

(c) CDS-Rule-K. (d) Comparison of best mechanisms.

Fig. 12.4 Network lifetime with and without dynamic local topology maintenance using the A3, EECDS, and CDS-Rule-K topology construction mechanisms in sparse networks.

nodes start reaching the energy threshold, triggering the local procedure more and more often. As time continues, the number of nodes triggering the local procedure increases making the advantage of CDS-Rule-K more appreciable over the A3 and EECDS counterparts.

12.3.2 Dense Networks

The simulation results of the energy- and DSR-based dynamic local technique in dense networks are presented in Figure 12.5. As before, the results improve with the density of the network, as more possibilities exist to reconnect the topology when more nodes are available. As the figure shows, in all cases the application of the dynamic local topology maintenance technique improves the network lifetime compared with the case of no topology maintenance. Further, the performance also improves over the case with sparse networks, as expected. Similar to the results in sparse networks, Figure 12.5(d) shows that the local technique based on the CDS-Rule-K algorithm outperforms the other two. The behavior is the same only that

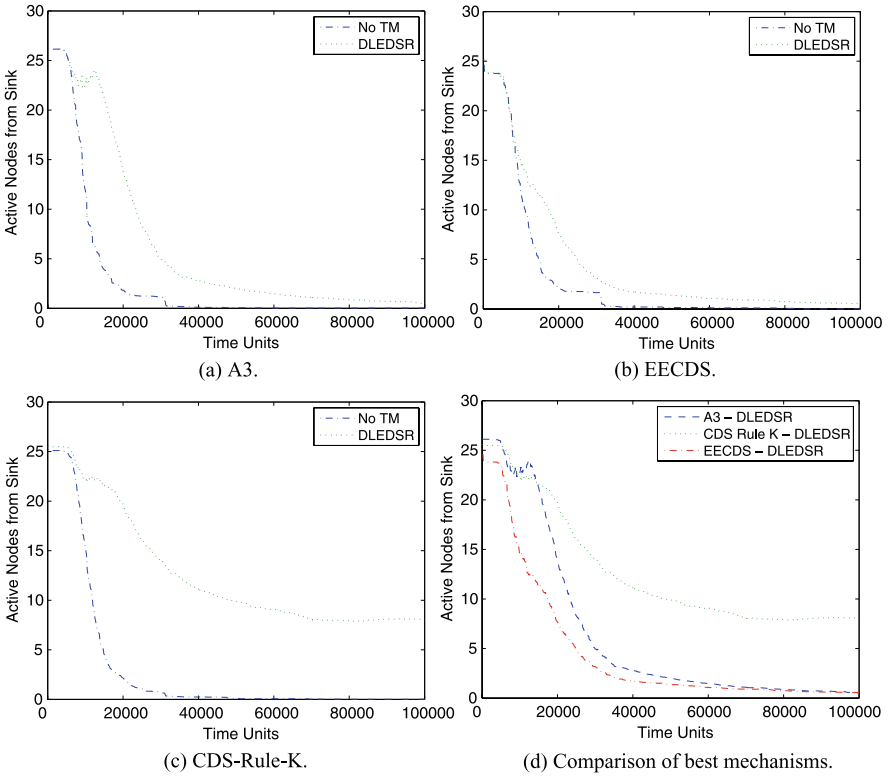


Fig. 12.5 Network lifetime with and without dynamic local topology maintenance using the A3, EECDS, and CDS-Rule-K topology construction mechanisms in dense networks.

with more nodes the performance is better. The CDS-Rule-K topology construction algorithm fits very well into this dynamic local strategy.

12.3.3 Other Dynamic Local Technique

There are several techniques available in the literature that perform dynamic local maintenance of the network topology without being explicitly categorized as dynamic local topology maintenance techniques. For example, in [148], the authors propose a fault tolerant topology maintenance methodology in which they select a set of backup nodes that will maintain backbone connectivity replacing nodes in case of failure. The selected backup nodes will turn on when a failure is detected in one of the active neighbors of a non-adjacent active node, creating an alternate route to reach the active node in case it was disconnected from the network. The algorithm works in a distributed manner based on neighbor lists, and it can handle $|S_a|$ failures, where S_a is the size of the set of active nodes.

Another dynamic local failure-based technique is presented in [36]. The work proposes to increase the transmission range of a neighbor of the failed node. The neighbor is selected either randomly or based on the remaining energy of the nodes.

In [21], the authors present an evaluation scheme for node failure in static wireless networks. The evaluation consists of periodical requests from each node to all the neighbors in range. The recipient of the request must execute a test task and return the result to the requester. If the result is correct and the message is received during an expected interval, then the node is considered available. The authors in [35] extended this evaluation procedure to mobile units, including recovery after failure.

The Geographical Adaptive Fidelity (GAF) algorithm introduced in [137] is another example of a dynamic local topology maintenance technique. GAF is based on the known observation that in dense networks multiple paths can be established between nodes, and therefore, there is the possibility of turning some nodes off without jeopardizing the network connectivity. GAF uses node location information and a virtual grid to introduce the concept of node equivalence in which all nodes within the same virtual grid square are considered equal in terms of their ability to route packets. Nodes in the same grid then coordinate among themselves to determine who will perform the routing function on their behalf and for how long. This negotiation is based on an application-dependent ranking procedure that chooses as the active node the one with the most energy. The active node then broadcast the amount of time it will be active, which is calculated based on the amount of remaining energy. Once the active node consumes half its energy, the nodes wake up and the negotiation procedure starts again. At that time, it is very unlikely that the recent active node will be selected again since it was active and the other ones sleeping, achieving the desired energy consumption balance.

The SPAN algorithm introduced in [20] is similar to GAF in the sense that each node decides to be part of the routing backbone based on the number of neighbors and its amount of energy. In SPAN, backbone nodes are called coordinators, and also rotate with time. One difference between the two algorithms is that SPAN does not need location information. SPAN is based on HELLO packets by which nodes know who their neighbors and coordinators are. If two neighbors of a non-coordinator node cannot reach each other either directly or through one or two coordinators, then the node should become a coordinator. In order to resolve contention problems when more than one node volunteers as a coordinator, SPAN delays the coordinator announcements by a time that considers the number of nodes among these neighbors that would be connected if the node were to become a coordinator, and the remaining energy in the node. Then, periodically, each coordinator checks if it should continue being a coordinator. In order to ensure fairness, the coordinator withdraws itself as a coordinator if every pair of neighbor nodes can reach each other via some other neighbors.

The Adaptive Fidelity Energy-Conserving Algorithm (AFECA) proposed in [136] is also very similar to GAF and SPAN. AFECA shares the same motivation and approach to topology maintenance of GAF and SPAN but uses a differ-

ent switching metric: neighbor density. Each node estimates its neighbor density by maintaining a list of the nodes it is able to hear. Then, it calculates its sleeping time using a proportional factor of the number of neighbors.

Chapter 13
Topology Maintenance Hybrid Techniques

13.1 Introduction

This chapter shows that combinations of static and dynamic topology maintenance techniques are possible, creating hybrid topology maintenance techniques. After describing and studying the performance of one of those possibilities, the rest of the chapter is devoted to compare and conclude about all the topology maintenance techniques presented in this part of the book.

13.2 Performance Evaluation of a Hybrid Global Topology Maintenance Technique

In this section one hybrid topology maintenance technique is presented and analyzed using the Atarraya simulation tool. The technique in question is a combination of a static global and a dynamic global technique, both triggered by either energy or time thresholds. The technique is static global in the sense that during the first topology construction process, it calculates all possible node-disjoint or shared-disjoint topologies and starts working as a regular static global technique, which rotates the Christmas trees every time one node reaches its energy or time threshold. As time goes by and the different trees are rotated, a new designated tree may not be able to provide its service because it has no communication with the sink node anymore. Under the normal static global technique, this tree would be eliminated from the rotation list and skipped, and the technique will continue working as long as there are valid trees remaining.

Under the hybrid technique, when the designated tree can no longer provide the service, it will trigger a dynamic global technique to find a new reduced topology, which may include in its reduced topology some nodes that had never been used before by any tree. The new, dynamically created, tree will work until the energy or time threshold is reached; at that time, the static technique will take over and

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

continue switching the pre-planned topologies. This procedure is repeated until no more static nor dynamic tress can be established in the network.

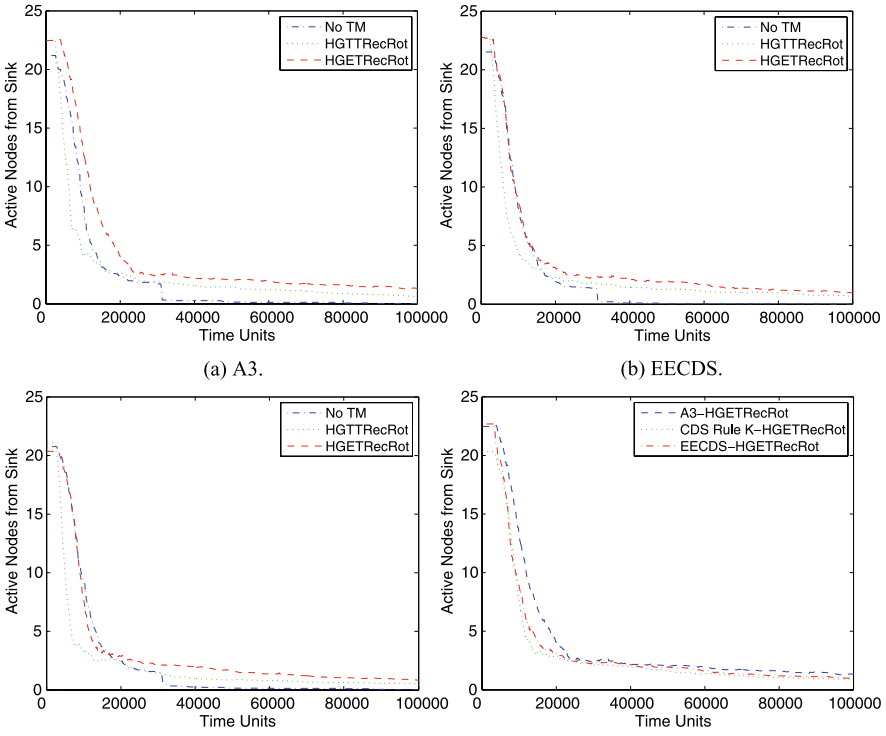
Using the same network parameters described in Table 10.1, and the same underlying topology construction mechanisms utilized in the performance evaluations included in Chapters 11 and 12, simulation experiments were carried out to assess the network lifetime of the hybrid global topology maintenance technique just described in sparse and dense networks. The following topology maintenance techniques and triggering criteria were included:

- **No Topology Maintenance – No TM:** The initial reduced topology works permanently until the sink detects that it does not have any more active nodes in range. This is the same termination policy for the rest of the algorithms.
- **Hybrid Global Time-based Topology Recreation Rotation – DGTTRecRot:** Every set time interval the topology maintenance algorithm rotates the active reduced topology for one of the preplanned ones. If the new pre-planned topology cannot provide the expected service (has no connection with the sink), the hybrid topology maintenance algorithm invokes the topology construction algorithm to create a new reduced topology on the fly.
- **Hybrid Global Energy-based Topology Recreation Rotation – DGETRecRot:** Every time a node reaches a critical energy threshold, the topology maintenance algorithm rotates the active reduced topology for one of the preplanned ones. If the new pre-planned topology cannot provide the expected service (has no connection with the sink), the hybrid topology maintenance algorithm invokes the topology construction algorithm to create a new reduced topology on the fly.

13.2.1 Sparse Networks

As in the case of the static and dynamic topology maintenance techniques presented in Chapters 11 and 12, the performance results of the hybrid global technique presented here shows no major improvement compared with the no topology maintenance option in sparse networks. In some cases, the technique increases the network lifetime and in some cases reduces it, but with small margins in both cases, emphasizing the conclusion reached before with other techniques that applying topology maintenance in sparse networks will not produce significant improvements in terms of network lifetime.

Figure 13.1(d) also emphasizes previous results. For example, it shows that the energy-based hybrid technique also outperforms the time-based technique. Also, the performance of the hybrid technique based on the A3 topology construction is consistent with previous results, showing its advantage although without impressive margins. Finally, compared with the static and dynamic counterparts, it seems that the hybrid technique improves the performance, but not considerably. (More on this in Section 13.3 later.)



(a) A3. (b) EECDS.

(c) CDS-Rule-K. (d) Comparison of best techniques.

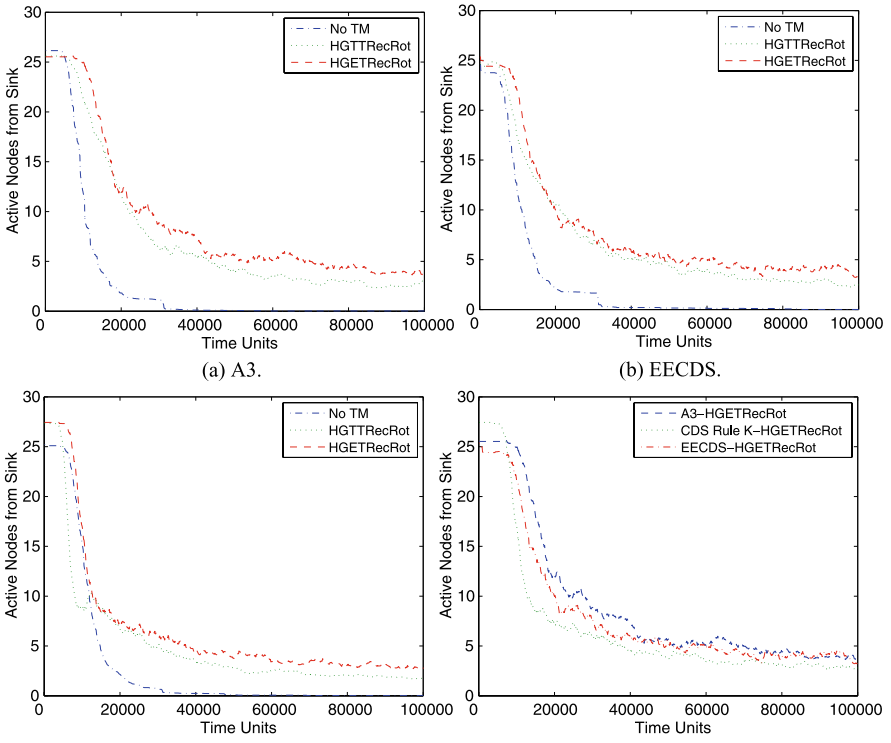
Fig. 13.1 Network lifetime with and without hybrid global topology maintenance techniques using the A3, EECDS, and CDS-Rule-K topology construction mechanisms in sparse networks.

13.2.2 Dense Networks

The same experiments were run in a more dense network, one with 100 nodes – twice the number of nodes utilized in the sparse scenario. The simulation results shown in Figure 13.2 clearly demonstrate the power of this technique. In all cases the hybrid global topology maintenance technique outperforms the no topology option by impressive margins. As in previous cases, Figure 13.1(d) shows that the hybrid technique with the A3 topology construction algorithm performs better than the technique with the EECDS and CDS-Rule-K algorithms. It is clear that, once the static portion of the technique exhausts the energy of the pre-planned trees, the dynamic technique still can find new trees, and therefore, run the network longer.

13.3 Comparison of Topology Maintenance Techniques

This section summarizes all the performance evaluation results presented thus far in this part of the book in order to provide general conclusions as to which technique is



(a) A3. (b) EECDS.

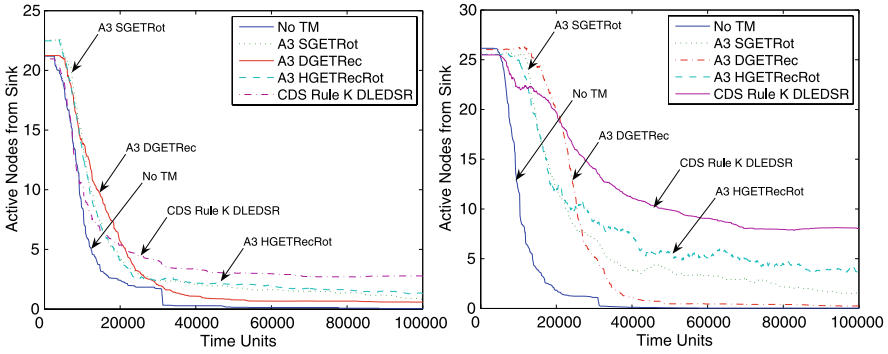
(c) CDS-Rule-K. (d) Comparison of best techniques.

Fig. 13.2 Network lifetime with and without hybrid global topology maintenance techniques using the A3, EECDS, and CDS-Rule-K topology construction mechanisms in dense networks.

better to use in sparse and dense networks. As such, this section is meant to answer questions, such as:

- What is the best static, dynamic and hybrid technique in sparse and dense networks?
- If I have a sparse or a dense network, what technique should I use?
- Which approach is better, global or local?
- What triggering criteria is better, energy-based or time-based?

All these questions can be answered by looking at the results presented in Figure 13.3, which includes the best performing techniques in sparse and dense networks using the same simulation parameters and scenarios outlined thus far. From the figure, three general conclusions can be easily drawn. First, consistent with all results presented so far, it can be seen that regardless of the network density, the scope of the technique (global or local), and type of technique (static, dynamic or hybrid) the energy-based techniques are the best performing ones. Second, the global techniques (either static, dynamic, or hybrid) perform better with the A3 topology construction algorithm. Finally, the local DSR-based technique based on the CDS-



(a) Best of all in sparse networks. (b) Best of all in dense networks.

Fig. 13.3 Best performing topology maintenance techniques in sparse and dense networks.

Rule-K topology construction algorithm outperforms the global techniques in both sparse and dense networks. Therefore, the energy-, DSR- and CDS-Rule-K-based dynamic local topology maintenance technique is the best performing technique in sparse and dense networks.

Another important conclusion about the behavior of these techniques is that dynamic global techniques tend to offer better performance longer than static and hybrid global techniques, but once the performance starts to decay, it decreases faster. The performance of static global techniques starts to decay sooner but the decrease is not as sharp, being able to live longer than dynamic global techniques over longer times. The hybrid global techniques are a good compromise. They have the same performance as static global techniques during the initial continuous time but their performance over time is considerably better than static or dynamic global techniques, with a smoother network lifetime decay. Finally, the DSR- and CDS-Rule-K-based dynamic local technique presents the worst performance at the beginning, meaning that nodes start to die faster, but shows the best performance of all over time.

More specific conclusions can be drawn looking at each figure individually. For example, from Figure 13.3(a), it can be seen that, again, topology maintenance techniques in sparse networks do not improve the lifetime of the network by an important margin compared with the no topology maintenance option. This is definitively not the case in dense networks, as it can be seen from Figure 13.3(b), where the positive effect of the topology maintenance techniques over the network lifetime is appreciable.

In conclusion, dynamic global techniques are the best option if a continuous operation of the entire network is needed over a specific amount of time, less than the time at which they start to die fast. If the application, on the other hand, requires the network to be alive longer even with fewer nodes and maybe with some areas not completely covered, then the dynamic local technique using the CDS-Rule-K topology construction mechanism is the best way to go. A good compromise in this spectrum of possibilities is the global hybrid technique, which provides better

performance than the local technique at the beginning (works with all the nodes longer), and an intermediate performance over time compared with the dynamic global and the local techniques.

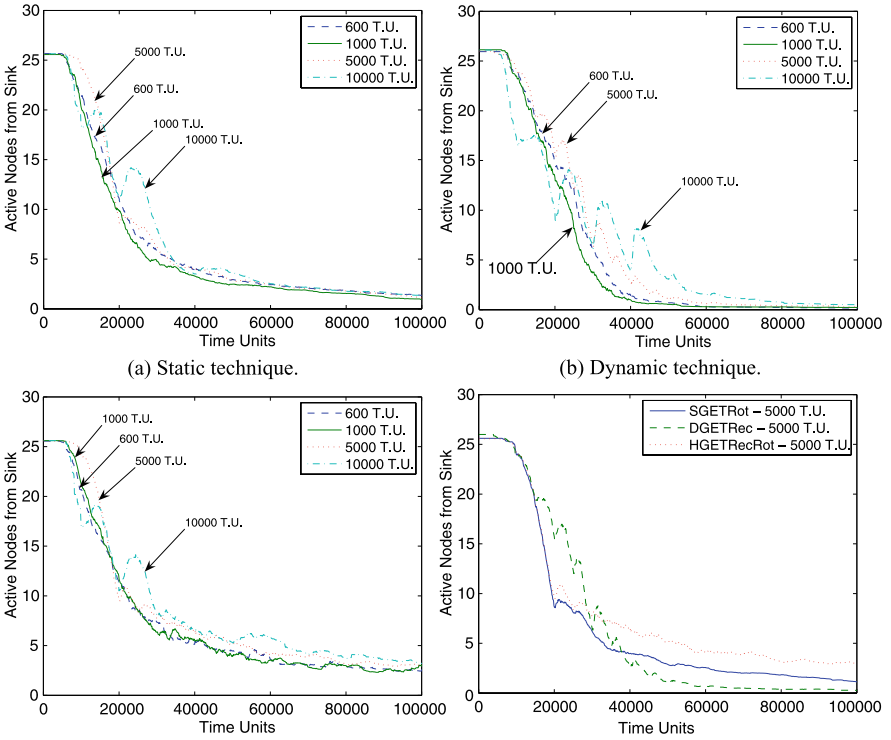
It is important to mention that these general and specific conclusions only represent one picture of the entire spectrum of possible scenarios. All these conclusions have been drawn from the simulation results of experiments run with 50 and 100 nodes, an energy threshold of 10%, a time threshold of 1000 time units, maximum of 3 reduced topologies in the case of static techniques, the A3, CDS-Rule-K, and EECDS topology construction algorithms, and the local DSR-based topology maintenance mechanism. In the next section, we make an attempt to extend these conclusions running additional experiments varying the energy and time thresholds, and the network density.

13.4 Sensitivity Analysis

Thus far, all simulation experiments have been performed using 50 or 100 nodes to represent sparse and dense network scenarios, respectively, and fixed energy and time thresholds, of 10% and 1000 time units, as defined in Table 10.1. This section is meant to provide more insights about the performance of the topology maintenance techniques under consideration when these parameters are varied. In what follows, simulation results are presented when the time threshold is varied from 600 to 1000 to 5000 to 10000 units; the energy threshold is changed from 5% to 10% to 25% to 50% of the remaining energy in the node; and the number of nodes is varied from 50 to 100 to 400 nodes. Experiments are run using static, dynamic, and hybrid global techniques using the A3 algorithm as the only topology construction algorithm, and the DSR-based dynamic local technique using the CDS-Rule-K algorithm only. Also, all experiments are run in dense networks only. These decisions are based on the superior performance of the global techniques based on the A3 algorithm, the superiority of the local technique based on the CDS-Rule-K algorithm, and the fact that, in most cases, topology maintenance is worth applying in dense networks only.

13.4.1 Time-Based Analysis

Figure 13.4 shows the network lifetime results for the static, dynamic, and hybrid global topology maintenance techniques with time thresholds of 600, 1000, 5000, and 10000 time units using the A3 topology construction algorithm. The figure provides insights on two fronts. First, it immediately tells about the impact of changing the time threshold in the network lifetime. As it can be observed, in all cases, the network lifetime seems to be mostly unaffected by the time threshold. However, the figures clearly show the effect of the threshold in the operation of the network. For example, the figures show the bumpy behavior of the techniques when the threshold is set at 10000 units. It is clear that the threshold time is big enough as to allow many nodes to die before it expires, and when it expires, the new topology is able to



(a) Static technique.

(b) Dynamic technique.

(c) Hybrid technique.

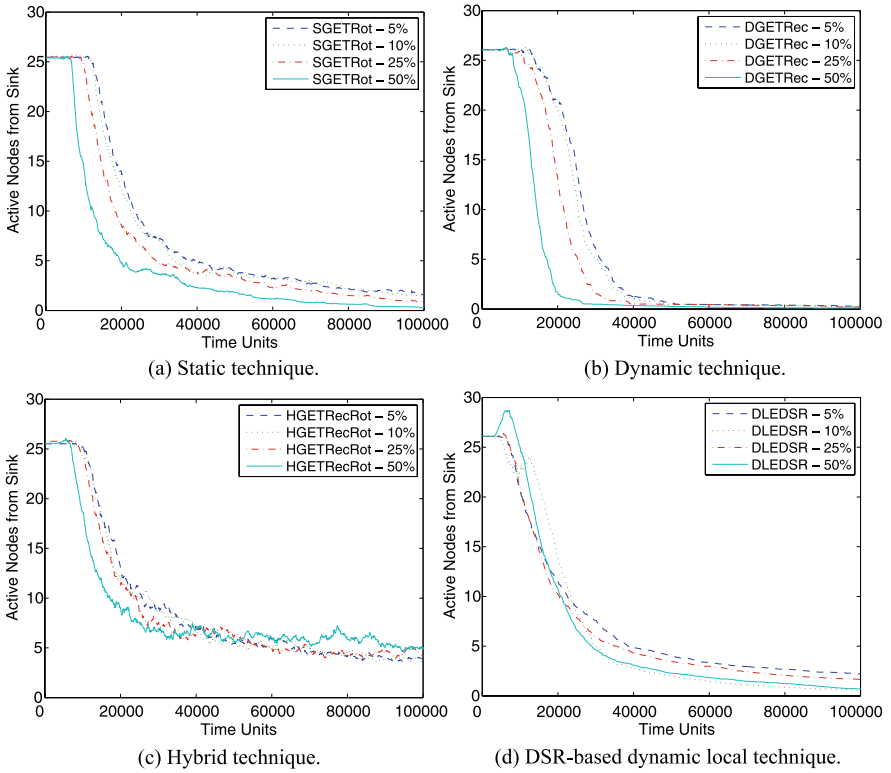
(d) Best of each.

Fig. 13.4 Time-based sensitivity analysis of network lifetime for static, dynamic, and hybrid global topology maintenance techniques using the A3 topology construction mechanisms in dense networks.

include more active nodes than the topology just replaced. On the other hand, this effect is not seen when a small threshold is used, such as in the case of the 600 time units threshold.

Second, the figure provides an overall comparison between static, dynamic, and hybrid global techniques. As it can be observed from Figure 13.4(d), a compromise between performance and smooth behavior is achieved when the time threshold is set to 5000 time units. As it can be seen, the best performing technique in each case is the one using this threshold of 5000 time units. At the same time, although the dynamic technique still presents some bumps, it is considerably better than the case with 10000 units. Of the three best techniques, the figure also shows that the hybrid technique is the most appropriate, presenting the same performance of the other techniques during the initial 15000 time units but decaying slower as the time increases.

In conclusion, it can be said that, as expected, increasing the time provides better performance, as the topology construction mechanism needs to be run fewer times, but increasing the time threshold too much also produces bumpy behavior. There-



(a) Static technique.

(b) Dynamic technique.

(c) Hybrid technique.

(d) DSR-based dynamic local technique.

Fig. 13.5 Energy-based sensitivity analysis of network lifetime for static, dynamic, and hybrid topology maintenance global techniques, and DSR-based dynamic local topology maintenance technique using the A3 topology construction mechanisms in dense networks.

fore, a good compromise is to set the threshold to a value close to the time at which the nodes are expected to start dying.

13.4.2 Energy-Based Analysis

In this part similar simulation results are provided by changing the energy threshold. Figure 13.5 shows the network lifetime results for the static, dynamic, and hybrid global topology maintenance techniques, and the DSR-based dynamic local topology maintenance technique with energy thresholds of 5%, 10%, 25%, and 50% of the node's remaining energy using the A3 topology construction algorithm. The figure immediately provides insights about the impact of changing the energy threshold in the network lifetime. As it can be observed, there is a general trend: the network lifetime improves with the reduction of the energy threshold. A smaller threshold corresponds to fewer runs of the topology construction mechanism, saving more energy.

Figure 13.6 provides an overall comparison between the techniques. The figure clearly shows that in this case, the hybrid technique is the best of all, starting its

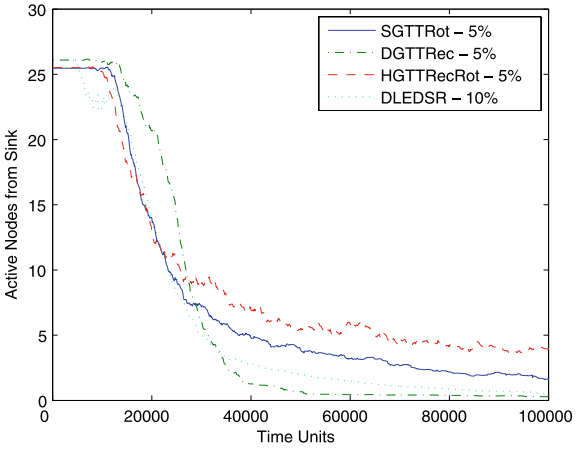
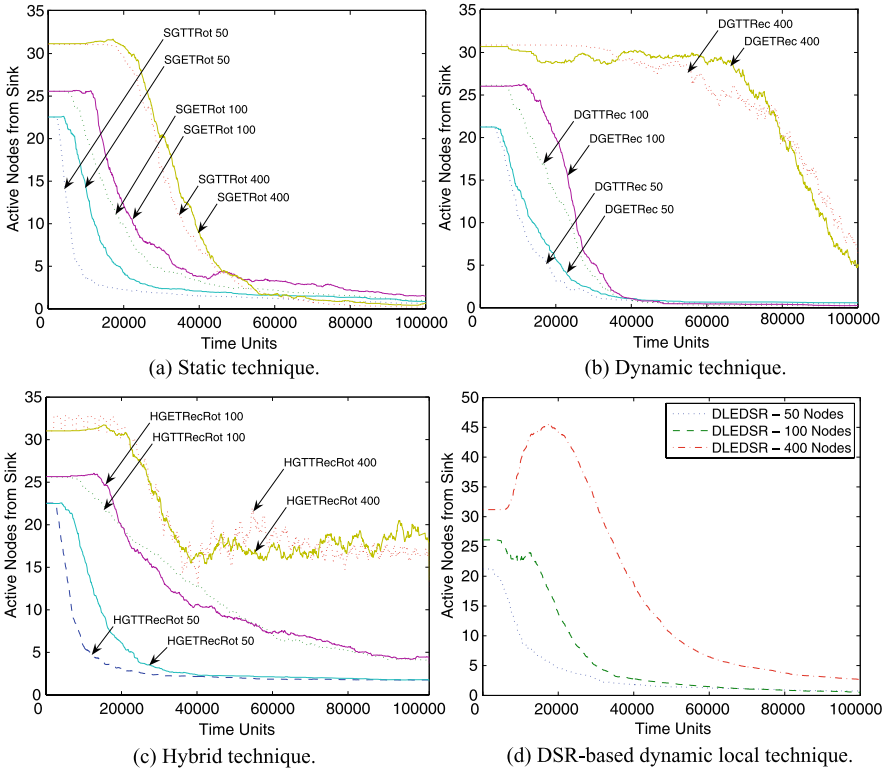


Fig. 13.6 Best performing techniques out of Figure 13.5.

performance decay almost at the same time as the rest of the policies, but outperforming them all over time. Given the results obtained in these last two sections, it can be concluded that hybrid techniques are the best overall, as they get the best of the static and dynamic techniques they are made of. It is important to mention that the local technique does not perform as well as the global ones because it is based on the A3 topology construction algorithm. As shown in Figure 12.5(c), if we had included here the performance of the local technique with the CDS-Rule-K algorithm, it would have been the best performing technique.

13.4.3 Density-Based Analysis

Finally, experiments were also carried out to assess the performance of the static, dynamic, and hybrid global techniques, and the dynamic DSR-based local technique in scenarios with different node densities using the A3 topology construction algorithm. Figure 13.7 shows the network lifetime results of the techniques under consideration with 50, 100, and 400 nodes using an energy threshold of 10% and a time threshold of 1000 time units. A general, and expected, trend can be easily observed from the figure: the network lifetime increases with the network density. As more nodes are added into the network, more disjoint topologies can be formed, and therefore the total lifetime can be increased. Also, more nodes increase the network wide energy resources. Another observation consistent with past results, is that the energy-based techniques outperform their time-based counterparts. Finally, Figure 13.5(d) shows the performance of the DSR-based energy-based dynamic local technique, which shows mixed results compared with the global techniques, but definitively, it shows its value in some instances. Recall that this local technique utilizes the A3 algorithm, and better performance results are obtained with the CDS-Rule-K algorithm.



(a) Static technique.

(b) Dynamic technique.

(c) Hybrid technique.

(d) DSR-based dynamic local technique.

Fig. 13.7 Node density-based sensitivity analysis of network lifetime for static, dynamic, and hybrid topology maintenance global techniques, and DSR-based dynamic local topology maintenance technique using the A3 topology construction mechanisms with time- and energy-based triggering criteria.

An interesting result is shown in Figure 13.7(c) where the effect of combining static and dynamic techniques can be clearly seen. At the beginning and during the first 20000 time units, the hybrid technique operates as the static one, as all topologies included in the rotation list can actually communicate with the sink node, with all their nodes participating in the topology. However, after 20000 time units, nodes start to die and the performance of the hybrid technique starts to decay following the slope of the static technique, as up to this point, the dynamic part of the technique has not been activated for the first time. It is at 40000 time units when the dynamic techniques start working, as some topologies in the static rotation list have no communication with the sink node any more. From this time until approximately 150000 time units (not shown in the figure) the performance of the hybrid technique is fairly stable. After that, most of the nodes die and the dynamic technique can no longer find more topologies.

Comparing these results, it can be concluded that increasing the network density can increase the network lifetime considerably. Also, among these techniques, the dynamic energy-based technique offers the best performance overall during a fairly

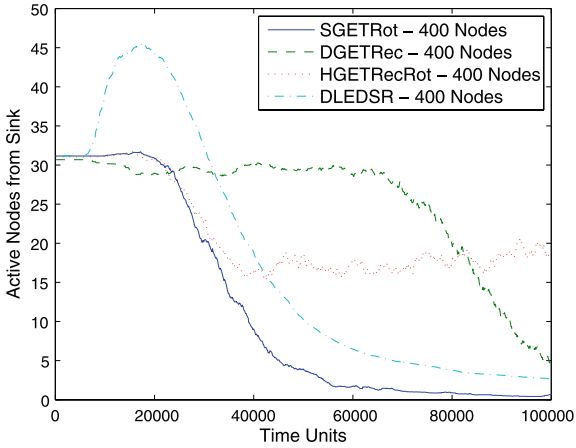


Fig. 13.8 Best performing techniques out of Figure 13.7.

large amount of time, after which the hybrid technique takes over maintaining an still very acceptable performance for almost twice the original time. Figure 13.8 includes the best performing techniques in each case, where this last conclusion is easily observed.

Appendix A
The Atarraya Simulator

A.1 Introduction

The main idea behind the creation of Atarraya – which means *fishnet* in Spanish – was to test the topology construction protocol named A3 that was being developed as part of this research. The software, as originally conceptualized, was very simple, rigid and tightly coupled with the A3 protocol. However, due to the fact that it was necessary to compare the performance of A3 against other known topology construction mechanisms, the design of the tool was not adequate. Therefore, the decision to build a more generic simulator, in which other topology construction algorithms could be plugged in, and have a single platform where to evaluate them all under the same conditions, was necessary. Then, the concept of topology control was also expanded to include topology maintenance algorithms, and several of these mechanisms were designed and included as well.

The final result is Atarraya: a generic, Java-based, event-driven simulator for topology control algorithms in wireless sensor networks. As with any simulation tool born out of a research effort, Atarraya is still in development; however, in its current state, it is an excellent tool not only for research, to develop and test new topology control algorithms, but also for teaching. Atarraya's graphical user interface shows how topology control protocols work, and how they shape topologies during their execution. In addition, Atarraya includes necessary mechanisms to experiment with classic theoretical results related to topology control in wireless sensor networks, such as the giant component experiment, calculation of the critical transmission range (CTR), calculation of the Minimum Spanning Tree of a graph, and others.

In this appendix the basics of Atarraya are presented along with its internal structure, so the reader knows how to develop and plug in new topology control algorithms and protocols, and a brief guide on how to use the tool. All explanations and descriptions included in this document are related to Atarraya's version 1.0, which is the version that was used to run all the experiments included in the book. Future versions and new features will be documented on the project's Website at http://www.csee.usf.edu/~labrador/Atarraya.

M.A. Labrador, P.M. Wightman,
Topology Control in Wireless Sensor Networks,

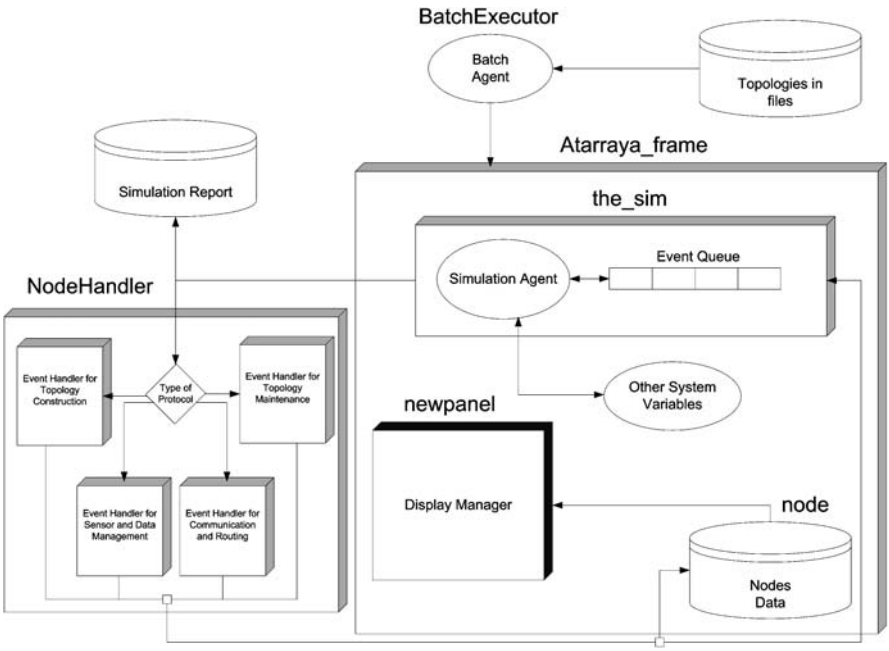


Fig. A.1 Atarraya's functional components.

A.2 Description of Atarraya's Internal Structure

In this section the internal structure of Atarraya is described. First, its main functional components and Atarraya's class tree are presented. Then, the structure of the protocols is described in more detail, including how they communicate with the main class and with other protocols, how to initialize the nodes, and how to handle protocol events.

A.2.1 Abstract Design and Functional Components

This section describes the main functional components of the simulator and how they interact among themselves. The functional components offer the "big picture" necessary to understand the critical components of Atarraya. Figure A.1 presents a global view of the internal structure of the simulator, which consists of the main simulator thread, the node handler, and the batch executor. The elements of this structure are described in this section.

A.2.1.1 The Main Simulator Thread – The *the_sim* Class

This is the core of the system. The simulator thread, defined in the class *the_sim*, is in charge of fetching the next event from the simulation event queue, and sending the event to the node handler for execution. An instance of this class is created by the

method StartSimulation() whenever a simulation is executed. This class contains the event queue, the simulation clock, the display manager, the database with the data about the nodes, and the simulation agent, which is in charge of storing the simulation results for the reports in the respective logs.

When an instance of the *the_sim* class is created, it is necessary to add the initial events to the queue before the thread is started. The first thing the thread will do once started is to check if there are any events in the Event Queue. If the thread is started without any events, it will consider that an error has occurred, and the simulation will be suspended.

Once the first events have been loaded into the queue, the simulator thread can get started. The thread starts a loop that will execute until one of the three termination conditions is true: there are no events in the queue, all the nodes have reached the final state in the topology control protocol, or the protocols have called for the end of the simulation (for example, the topology maintenance protocol has found that the sink has no more neighbors, so the network is dead). If the first condition occurs and the simulator has not been notified that the protocols finished execution, it means that there was an error during the simulation, and it will be notified on the simulation report.

In the loop, the first thing the thread does is to verify if the event is valid. If so, the event will be registered (if this option was selected by the user), the simulation clock will be updated, and the event will be sent to the *NodeHandler*. There, the event will be delivered to the appropriate *EventHandler* according to the respective protocol the event belongs to. Once the event is executed, the simulator will go back to the loop and start again the process. The simulator updates the clock with the execution time of the events based on the fact that all the events in the queue are sorted by their projected execution time, so there is no such thing like a trip to the past.

Once the simulator breaks the loop by any of the finalization conditions mentioned above, the thread goes to the report construction section, saving all the events and statistics, as selected by the user. This section also takes into account whether or not the simulation is part of a batch execution, in which case all the data from all previous executions is kept until the last one finishes. All this information is stored in data structures that are stored in the report files after the simulation is finished. Reading this section of the code will provide the user with information about all the options for each configuration of report in both single and batch simulation cases. Once the simulation and the report building section finish, the thread ends too.

In the current version of the simulator, just one simulator thread can run at a time because there is only one data structure to store the topology, which is localized in the *atarraya_frame* class. Individual instances of the data structure running several simulations in parallel will consume all the resources of the Java virtual machine, especially if the network topologies are big.

A.2.1.2 The Protocol Manager – The *NodeHandler* Class

This class is in charge of defining the protocols to be used in the simulation and routing the event to the appropriate protocol once received from the simulation thread.

The *NodeHandler* class defines the four possible protocols that a node can have running during a simulation: Topology construction, topology maintenance, sensor-data management, and communication-routing protocols. Given that there are different algorithms for each type of protocol, the main purpose of this class is to make that selection transparent to the rest of the simulator, so that no detail about the selection is required in order to execute the simulation. When a simulation is started, this class creates the instances of the selected protocols in each of the four different categories. The simulation thread sends the next event from the event queue to the *NodeHandler* class. Once the event is received, it is routed to the appropriate protocol based on the protocol identifier included in the event.

A.2.1.3 The Multiple Operation Thread – The *BatchExecutor* Class

The main purpose of the BatchExecutor thread is to perform operations that require multiple executions, such as creating a set of topologies, performing a large number of simulations, and the Giant Component test. Since these operations are run on a thread independent from the main one, the graphical user interface does not freeze while these operations are being executed, which allows for the interaction between the user and the simulator even while some of these operations are running in the background. This class is instantiated whenever one of the mentioned operations is started.

A.2.1.4 The Display Manager – The *newpanel* Class

The display manager, or *newpanel* class, is the one in charge of the graphical representation of the topologies. The heart of this class is the override of the Paint method of this class that extends a Panel class. All the painting options for the topology are defined in this method. The other methods perform minor but necessary actions like obtaining information about the options, providing coordinates from the deployment area, etc. This class was defined as a private class of the *atarraya_frame* class so it can have direct access to the topology data structure.

Atarraya provides several options for topology visualization, which can be seen in more detail in the visualization options in Figure A.12. The most relevant visualization options are the following:

- MaxPower Topology: This is the original view of the topology with all nodes transmitting at full power, and all the links that their unit disks provide.
- Single selected network configuration or tree: In this view the user defines which of the Virtual Network Infrastructures (VNI) he or she wants to see. More on VNI later in Section A.4.3. The default configuration is Black in most protocols.
- All network configurations: If several configurations are defined in a certain topology, this view allows the user to see all of them and appreciate the differences between them.
- Active network configuration: Each node is assumed to be able to maintain several VNIs, but use only one at a time. This view allows the user to see in real-time in which network configurations the nodes of the topology are.

A.2.2 Atarraya's Class Tree

In this section the class tree of Atarraya is described and a brief explanation of the structure and mission of the current internal structure of the application is provided. The classes in Atarraya are organized in three packages:

- *Atarraya*: the main functional elements are stored here, such as the main frame, the simulation agent, and the display manager.
- *Atarraya.element*: This package contains the classes that model the data structures, like the node, VNI, routing table, etc.
- *Atarraya.event*: This package contains the classes related to the protocols and the definition of the event queue.

A.2.2.1 The *Atarraya* Package

The *Atarraya* package is the main package of the simulator. It contains the following classes:

- *Main* class: This is the launcher of the simulator. It invokes the title frame and the main frame.
- *atarraya_frame* class: This is the main class of the simulator. In this class we find the definition of the graphical user interface and the simulator core.
- *newpanel* private class: This class defines the operation related to the visualization panel for the topologies: painting, selection of coordinates, selection of nodes, grid, etc. It is a private class of the *atarraya_frame* class so the *newpanel* class has direct access to the data structures.
- *the_sim* private class: This class defines the structure of the thread that simulates a scenario; in other words, this class is the simulation executor. It was made private also to preserve the direct access to the data structures.
- *BatchExecutor* class: This class is in charge of executing operations that involve multiple scenarios, being that create multiple topologies, or simulate multiple scenarios. The advantage of using a separate class is that it creates a different thread that freezes the main frame while executing.
- *constants* interface: This interface defines the standard values for multiple variables. Given that many classes must share a set of standard values, the use of the *constant* interface allows this without having to define identical variables on each class.
- *FrameLogo* class: Initial frame with the logo of the simulator.
- *AboutFrame* class: Frame that contains the "About us" message.

A.2.2.2 The *Atarraya.element* Package

This package contains the classes that define the elements that will be used in the simulator for storing information. These classes are:

- *node* class: This class represents all the information about a single node and all the operations that can be performed on the individual nodes.
- *register* class: This class contains the information that the simulator has about the neighbors of a node. There is a difference between the information that the

node has about its neighbors and the information that the simulator has about those nodes. For example, the simulator needs to know the exact position of each node, but maybe the node does not have the ability to know the position of its neighbors.

- *candidate* class: This class contains the information that the node has about its own neighbors. This class has two data structures: candidate list and children list. Both are general use data structures on the node.
- *NodeNetworkConf* class: Based on the assumption that each node may have different VNI, this class represents everything the node needs to know from its current network: list of neighbors, list of gateways, sink's address, routing table, etc. A node can have as many different VNI as desired. By default, the maximum number of VNI is 5 and every node is in the VNI 0 and will remain there until something else is defined.
- *NodeSensingConf* class: This class is similar to the *NodeNetworkConf* class, but applied to the sensing device. Each node may have several sensing devices or different configurations. This class has not been fully implemented.
- *routing_table* class: This class defines the data structure and methods of the routing table of a VNI of a node. The main structure is a vector of *routing_table_register* elements. Examples of functionalities of this class are obtaining the appropriate gateway node for sending a packet through the best route, adding routes, storing message sequence numbers, etc.
- *routing_table_register* class: This class defines the format of a register of the routing table. It includes the address of the destination node, the ID of the next hop and routing metric, and the vector for storing the history of message sequence numbers, which are important for routing algorithms.
- *TMStructConfList* class: Some topology maintenance protocols require a great amount of information, especially those that can run several processes in parallel. This class models a data structure that implements a list of the independent *TMConfStructure* instances that a node is handling at a certain moment in time. The concept is similar to the one of the VNI, but just applies to the topology maintenance protocol.
- *TMConfStructure* class: This class defines the information that a node has about a single topology maintenance process, such as the identifier, the node's state on the topology maintenance process, and a routing table that could be used when specific paths must be restored.
- *pair* class: A class that defines a pair of integer values.
- *trio* class: A class that defines a set of three integer values.
- *edge* class: A class that defines an edge (source, destination, and weight). This class is used to calculate the MST of a graph.

A.2.2.3 The *Atarraya.event* Package

This package contains the classes that define the structure of the events, the event queue, and the event handlers that will execute the events accordingly. The main classes are:

- *event_sim* class: This class defines the information of an event: source, destination, type of event, embedded information, configuration, layer that generated the event, etc. This class contains all the information that the event handler requires to execute the event properly.
- *eventQueue* class: This class defines the queue of events that Atarraya uses during the execution of a simulation. Events are added to the queue in an organized way, based on the projected execution time of the event, so the event on the head of the queue is always the closest to the present time.
- *EventHandlerxxx* class: This is a family of classes that define the protocols. Each class has to define some initialization operations for the nodes, how the events will affect the state of the nodes, the data structures, other events that get triggered as a consequence of the occurrence of one event, etc. A class of this type needs to be designed if a new topology control algorithm is to be included in the simulator, or if an existing algorithm is to be modified. In its current version, Atarraya supports four types of protocols: Topology Construction, Topology Maintenance, Sensor-Data Management, and Communication-routing protocols. The way these protocols communicate is by generating events of each other's type.
- *NodeHandler* class: This class defines a data structure that holds the selected options for the four types of protocols. Any new protocol added to Atarraya must be included in the existing list that this class contains.

A.3 Protocol Structure and Design – The *EventHandler* Class

This section introduces the design and structure of the protocols in Atarraya. The EventHandler class is the one that models the structure of a protocol in Atarraya. The next subsections describe the types of events that a protocol in Atarraya can model, how states are labeled, how each protocol communicates with the *atarraya_frame* class, how protocols interact with each other, how nodes are initialized, and how the simulator handles events.

A.3.1 Simulation Events

Given that Atarraya is an event-driven simulator, everything that happens during a simulation is an event, so protocols must be defined in terms of cause-effect when certain event occurs. Each of these types of events triggers some internal actions in the node that might modify its status, data structures, etc., and could also cause the generation of new events in the future. The most common examples of events in a protocol are:

A.3.1.1 Sending Messages

When a node intends to send a message, it may be addressed to a specific node (unicast) or it may be intended for every neighbor within range (broadcast). Regardless,

the method that a node needs to call on in order to send a message is broadcast. The parameters are the time at which the packet is received, the current time, the id of the sender node, the id of the receiver node (if it is a unicast message), the type of message, the payload, and the VNI corresponding to this package. In general, a message of this kind is assumed to be of the same type as that of the protocol that contains it, which explains why there is no specification of the protocol's type. More data can be included in this message, like the first sender of the message or source, and the final destination of the packet, in case it is a message that will travel through multiple hops.

```
broadcast(temp_clock+getRandom(MAX_TX_DELAY_RANDOM),
temp_clock, sender, -1, HELLO, temp_data,temp_vni);
```

The method broadcast is in charge of generating the reception events in the neighbors within communication range of the sender node, if the recipients and the sender node are active. The method broadcast is defined on the *atarraya_frame* class.

A.3.1.2 Receiving Messages

When a node receives a message, it calls the event determined by the type of message. If the message is supposed to be a unicast transmission, the node verifies if the id of the destination node matches its own. If that is the case it continues with the execution of the algorithm. If the receiver and the destination do not match, the node ignores the packet. Now, if the message was designed to be a broadcast transmission or the node needs to snoop in the packets not addressed to itself, it can ignore the receiver destination verification and just continue with the execution of the protocol.

```
case HELLO:
if(receiver == destination || destination == -1){
... //If the message was a unicast and the receiver was the
      destination or the packet was a broadcast
}else{
... //If the packet was addressed to another node and this
      is snooping
}
break;
```

A.3.1.3 Programming a Timeout

Sometimes a node needs to wait some time in order to perform a certain action. The definition of a timer is crucial for this type of operation. A timer in Atarraya is an event that a node programs addressed to itself in the future. The parameters are very similar to the ones provided to the *broadcast* method, with the difference that here the parameters of a new event, that will be included directly in the simulation queue, are also specified. Since this is a reflective event, the sender and the receiver have the same value. Also, in the declaration it is necessary to specify the target protocol of this event in the variable type. In the example, the node sender is programming

itself an event of the type *PARENT_RECOG_TIME_OUT*, that will be executed in *TIMEOUT_DELAY* time units.

```
pushEvent(new event_sim(temp_clock+TIMEOUT_DELAY, temp_clock,
sender, sender,PARENT_RECOG_TIME_OUT, "",temp_vni,type));
```

A.3.1.4 Invalidating a Programmed Event

A node programs events in the future without knowing what will really happen between the current time and the future event. For example, a node can be programmed to send a message in the future, but for some reason it may also be put to sleep before it can send the message. Since that particular event will not occur, it has to be taken out from the simulation queue, where they are waiting to be executed. Atarraya provides the methods *InvalidateAllEvents* in order to guarantee that a node can cancel events that should not happen. In the example, the sender node eliminates all the events of the current protocol, referent to the VNI *temp_tree* from the current time forward.

```
InvalidateAllEventsFromIDFromTimeTOfTypeTy(sender,
temp_clock, type,temp_vni);
```

A.3.2 State Labels

In general, a good number of topology construction protocols use node states to represent the evolution of the protocol. In Atarraya, nodes can be in any of the following four states: Initial, Active, Inactive, and Sleeping states. The definitions of these four states are included in the *Node_Handler* class. The values defined as the parameters are usually defined in the *constants* interface, and they are all positive integer values.

```
tc_protocol_handler.setLabels(S_INITIAL_STATE, S_ACTIVE,
S_SLEEP, S_SLEEP);
```

Given that most topology control protocols implemented in Atarraya are completely distributed, the sink cannot call the end of the protocol because it has no information about the state of all the nodes. That is the reason why Atarraya knows that a protocol has finished when all the nodes have reached the final state. Each topology control protocol can define which states are selected as the final states. This is done in the method *CheckIfDesiredFinalState(int s)* that is defined in every *EventHandler*, which is invoked in the *atarraya_frame* class when the simulation agent is trying to verify if the topology control algorithm is finished. In the following example, the protocol is selecting the active and the inactive states as the final states of the nodes.

```
public boolean CheckIfDesiredFinalState(int s){
        if(s==active || s==inactive)
            return true;
        return false;    }
```

Atarraya stops whenever the nodes of the topology are in any of the selected states, no matter if there are still events in the queue.

A.3.3 Communication with the atarraya_frame Class

Each protocol receives a reference to the instance of Atarraya's main frame, as defined in the *NodeHandler*. This reference allows the protocol to have access to variables from the main class. In order to access the variables from the simulator, the protocol needs to use the method *father.getVariable(int code)*, where the parameter *code* is a defined label for the sets of variables that can be accessed. This list can be found in the *constants* interface, and in the *atarraya_frame* class where the *getVariable()* method is defined. For example, if we need to know how many nodes are in the topology (including the sink nodes), the following line returns this value:

```
tam = (int)father.getVariable(NUMPOINTS)
```

When the protocol needs to get information about a node or modify it, the method to use is *getNode(int id)*, where *id* is the unique id of the node. In order to set node *i* in the initial state of the protocol in the VNI _vniID, the following line can be used:

```
getNode(i).setState(initial,_vniID)
```

A.3.4 Interaction with Other Protocols

There will always be some level of communication between protocols. For example, inter-protocol communication is needed to avoid situations like one node wanting to send a data message without having a route to the sink. Given that in Atarraya every event in the simulation goes to the same queue, it is necessary to determine to which protocol it must send the event to. Each type of protocol has its own identifier label, which is included in the event definition. This allows the *Node_Handler* to send the event to the appropriate protocol.

One of the premises of Atarraya is to create modular protocols that can be used in as many combinations as possible with the other protocols. Accordingly, protocols in Atarraya can only generate events in other protocols. For example, once a node reaches the final state of its topology construction algorithm, it can notify the topology maintenance protocol to start the maintenance procedure. Most of the times these inter-protocol events are meant to initiate or stop certain activity, so the protocol and how it works internally are completely independent, but the other protocols can decide the starting points. The following example illustrates a topology construction protocol when it invokes the topology maintenance protocol.

```
pushEvent(new event_sim(temp_clock+DELTA_TIME, temp_clock,
receiver, receiver, INIT_EVENT, "",temp_tree,TM_PROTOCOL));
```

A.3.5 Initialization of Nodes and the Initial Events – The init_nodes and the initial_event Methods

The *init_nodes(int vni)* method is used to set the nodes ready to start the execution of the simulation. Nodes are set to their initial states, and any previously defined events

regarding other protocols and all necessary variables are set to their default values. This method is invoked in the *StartSimulation* method in the *atarraya_frame* class, for all nodes, including the sink. The following code is an example of a *init_nodes* routine, in which every node is set to its initial state, every state label is defined, any existent programmed event in the queue is cancelled, and the execution of the topology maintenance and sensor and data management protocols are reset.

```
public void init_nodes(int _vniID){
 tam = (int)father.getVariable(NUMPOINTS);
 _clock = father.getVariable(CLOCK);
 temp = 0;

 for(i=0;i<tam;i++){
 getNode(i).setState(initial,_vniID);
 getNode(i).SetInfrastructureStarted(_vniID,true);
 getNode(i).defineLabels(initial,active, inactive, sleeping);

 if(TM_Selected){
 //stop all future event from the TM protocol
 pushEvent(new event_sim(_clock+getRandom(PROCESSING_DELAY),
 _clock, i, i, RESET_TM_PROTOCOL, "",_vniID,TM_PROTOCOL));
 }

 if(SD_Selected){
 //stop all future event from the sensor-data protocol
 pushEvent(new event_sim(_clock+getRandom(PROCESSING_DELAY),
 _clock, i, i, RESET_QUERY_SENSOR, "",_vniID,SENSOR_PROTOCOL));
 }

 }

}
```

The *initial_event(int _id, int _vniID)* method is used to define the first events to be included in the queue, before the simulation agent is started. Remember that if the simulation agent finds an empty queue it considers that the simulation has finished in an incorrect way. This method needs two parameters: the ID of the node that will perform the first event, and the VNI ID. This method is also invoked in the *StartSimulation()* method in the *atarraya_frame* class, but only for the sink nodes. If no sink nodes are defined in your topology, make sure that events are included in the queue using the *init_nodes(_vniID)* method.

A.3.6 The HandleEvent Method

This method is the core of the protocol, as it defines the actions taken by the protocol when an event occurs. The unique parameter that this method receives is the event taken from the event queue.

The events are classified based on an event label. Each protocol defines a set of labels for all the events that it uses. These labels are defined in the *constants* interface. The first action taken by the *HandleEvent* method is to recover all the

fields from the event and store them in temporary variables. Depending on the nature of the protocol, the classification of the events can be done in different ways: Label-then-State or State-then-Label. In the first case, the most important information is the label of the event, which becomes the main classification factor. Once the label is found, the code inside determines if the state of the node is important or not for the execution of the actions associated with the event. In the second case, the most important information is the state of the node. This methodology is useful when there are not many types of events but each type is interpreted differently based on the state of the node.

The following code shows the *HandleEvent* method from the example protocol presented in this section, the SimpleTree.

```
public void HandleEvent(event_sim e){

 int code = e.getCode();
 int sender = e.getSender();
 int source = e.getSource();
 int final_destination = e.getFinalDestination();
 int receiver = e.getReceiver();
 int destination = e.getDestination();
 double temp_clock = e.getTime();
 String temp_data = e.getData();
 int temp_vni = e.getTree();

switch(code){

case INIT_NODE:
     init_node(temp_vni, receiver);
        break;

     case INIT_EVENT:
      initial_event(receiver,temp_vni);
         break;

     /*
     * This event is when a node will start
     * sending Hello message to its neighbors
     */
    case SEND_HELLO:

   //The node will clean the candidates from
   //the neighbor's hellos messages
   getNode(sender).cleanCandidates();

    getNode(sender).setState(S_IN_SEARCH,temp_vni);

   //Sending Hello message to all neighbors
   broadcast(temp_clock+getRandom(MAX_TX_DELAY_RANDOM),
   temp_clock, sender,-1, RECEIVE_HELLO, temp_data,temp_vni);

   //Final timeout for evaluating candidates and adopt
   //children nodes
```

```
      pushEvent(new event_float(temp_clock+TIMEOUT_DELAY,
      temp_clock, sender, sender, LISTEN_4_REPLY_TIME_OUT,
      "",temp_vni,type));

        break;

       /*
       * This event is when a node received a Hello message
       * from its parent
       */
       case RECEIVE_HELLO:

          //If the node does not have a parent
          //in the current tree...
          if(!getNode(receiver).isCovered(temp_vni)){

  //Decompress the data in the message
  temp_data_array = temp_data.split("@");
  try{
  //The level of the parent is coming in the data
          temp_data_int = Integer.parseInt(temp_data_array[0]);

          //The sink address of this VNI
          temp_data_int2 = Integer.parseInt(temp_data_array[1]);
     }catch(Exception ex){ex.printStackTrace();}

          //Change the state of the node from the initial state
     getNode(receiver).setState(S_VISITED,temp_vni);

     //Define the parent in the current tree and
     //the sink address
     getNode(receiver).setParent(temp_vni,sender,sender,
     temp_data_int2);

     // Define the level
     getNode(receiver).setLevel(temp_data_int+1);

     // Schedules the Broadcast of the Reply message
     pushEvent(new event_float(temp_clock
     +getRandom(PROCESSING_DELAY), temp_clock,receiver, receiver,
     SEND_REPLY, "",temp_vni,type));

     //If it is not a parent after TIMEOUT_NO_PARENT units,
     // it will go into S_SLEEPING mode
     pushEvent(new event_float(temp_clock+TIMEOUT_NO_PARENT,
     temp_clock, receiver, receiver, END_TIMEOUT_NO_PARENT,
     "",temp_vni,type));
          }


          break;

  /*
      * This event is when the timeout for listening to other
```

```
     * neighbors finishes, and the node can send its own reply
     * to the parent.
     */
  case SEND_REPLY:

       //Parent's ID
       temp_ID = getNode(sender).getParentID(temp_vni);

  //The metric can be whatever the user needs it to be:
  //energy, ID, etc.
       metric = getNode(sender).getMetric(temp_ID);

  //You can use thresholds to limit responses...
  //For example, if energy is low, do not answer
        //y=java.lang.Math.random();
        y=0;
        if(metric > y){
        //Determine the metric to send
          temp_data = "" + metric;

          //Broadcast the message
          broadcast(temp_clock+getRandom(MAX_TX_DELAY_RANDOM),
          temp_clock, receiver, temp_ID, RECEIVE_REPLY,
          temp_data,temp_vni);
     }
     break;

      /*
       * This event is when a node received a reply
       * from a Hello message from its children or neighbors.
       */
     case RECEIVE_REPLY:

     //If the parent is the one that receives this message
     if(receiver == destination){

         //Obtaining the neighbor identifier in the node
         temp_ID = getNode(receiver).getNeighborIndex(sender);

         //Obtaining the metric that the node sent
         try{
       metric = Double.parseDouble(temp_data);
         }catch(Exception ex){ex.printStackTrace();}

         //Include a new neighbor from which the node heard
         //a hello message
         getNode(receiver).addCandidate(new candidate(temp_ID,
         sender,metric));

      }

      break;
```

```
/*
       * This event is when a parent node's timeout
       * to select children finished.
       */
case LISTEN_4_REPLY_TIME_OUT:
      i=0;
          sw=true;
          temp=0;
          temp_data="";

          //Number of candidates
          temp2=getNode(sender).getNumCandidates();

          if(temp2>0){

           //Changing the state of the node to be parent
             getNode(sender).setState(S_PARENT,temp_vni);
             getNode(sender).setParent(temp_vni, true);

//Initiate the TM protocol only on the active
//nodes of the topology!!
          if(TM_Selected){
          pushEvent(new event_sim(temp_clock+DELTA_TIME,
          temp_clock, receiver, receiver, INIT_EVENT,
          "",temp_vni,TM_PROTOCOL));
         }

          //Initiate the sensor querying protocol only
          //on the active nodes of the topology!!
          if(SD_Selected){
           pushEvent(new event_sim(temp_clock+DELTA_TIME,
           temp_clock, receiver, receiver, INIT_EVENT,
           "",temp_vni,SENSOR_PROTOCOL));
          }

          //Initiate the COMM  protocol only on the active
          // nodes of the topology!!
          if(COMM_Selected){
           pushEvent(new event_sim(temp_clock+DELTA_TIME,
           temp_clock, receiver, receiver, INIT_EVENT,
           "",temp_vni,COMM_PROTOCOL));
          }

          for(i=0;i<temp2;i++){

           //Get the i th candidate
              temp_cand = getNode(sender).getCandidate(i);

              //If the ID of the candidate is greater than 20,
              //then it will be a child!!
              if(temp_cand.getID()>20){
               //Adding the new child
                  getNode(sender).addChild(getNode(sender).
                  getNeighbor(temp_cand.getIndex()));
```

```
                    //Sending the packet
broadcast(temp_clock+getRandom(MAX_TX_DELAY_RANDOM),
temp_clock, sender, temp_cand.getID(),
ACCEPTANCE_MESSAGE, temp_data,temp_vni);
                    }

                }

            }

            break;

        /*
         * This event is when a node is accepted by the sender
         * node and it gets permission to start its own branch
         * of the tree.
         */
case ACCEPTANCE_MESSAGE:
        temp=0;
           if(sender==getNode(receiver).getParentID(temp_vni)&&
           receiver == destination){

            pushEvent(new event_float(temp_clock
            +getRandom(MAX_TX_DELAY_RANDOM),
            temp_clock, receiver, -1, SEND_HELLO,
            ""+getNode(receiver).getLevel()+"@"
            +getNode(receiver).getSinkAddress(temp_vni),
            temp_vni,type));

           }
           break;

        /*
         * This event is when a node did not received
         * an ACCEPTANCE_MESSAGE.
         */

case END_TIMEOUT_NO_PARENT:

          if(!getNode(sender).getState(temp_vni) == S_VISITED){
              frame_repaint();
              getNode(sender).setState(S_SLEEPING,temp_vni);
          }

          break;

        }

}
}
```

A.3.7 SimpleTree: An Example of a Topology Construction Protocol

In order to illustrate all these concepts, a simple example of a topology control algorithm follows. The selected example is a hierarchical topology construction protocol based on the growing a tree technique (see Section 8.2.1). The idea is to illustrate some common message exchange sequences, the use of timeouts, and how the modification of the status of the node modifies the execution of the protocol. The protocol works based on the following assumptions:

- The growing a tree protocol leaves every node active, except those with ID number less than 20.
- The nodes have no information about their position and have no list of neighbors.
- Every node starts in a unvisited state.
- The sink is the initiator of the process.
- The protocol ends when every node is in active mode or in sleeping mode.
- The initial topology is connected.

The protocol, step by step, works as follows:

- The sink node initiates the protocol sending a *Hello* message. It includes its address and level (number of hops from the sink), which in the case of the sink it is equal to 0. The sender node programs a timeout to stop listening for answers from its neighbors.
- All *unvisited* nodes on the transmission range of the sender node that received the *Hello* message answer back with a *Reply* message, set the sender as their default gateway, and change their status to *In-Process* mode. The receiver nodes at the same time set a timeout in case they do not receive an acknowledgment from their default gateway (the sender node).
- Once the timeout of the sender expires, this node checks the list of neighbors that answered back with the *Reply* message. If the sender node did not receive any answer back, it turns itself off and changes its status into a *Sleeping* mode. If the sender node received at least one answer back, it goes into *Active* mode and sends an unicast message to each one of them in order to let them know they were selected. All neighbors except those with ID number less than 20 will receive this unicast message.
- Once a node receives the confirmation message from its default gateway, it waits a random amount of time and sends its own *Hello* message to discover new *unvisited* nodes.
- If the timeout of a node in *In-Process* mode expires, it means that the node was not selected, so it turns itself off and goes into *Sleeping* mode.
- Once the node finishes its process and ends up in an *Active* mode, it starts the topology maintenance, sensor and data management, and communication protocols.

Although describing a protocol in words is useful for understanding its operation, they are more rigorously defined by Finite State Machines, especially in those

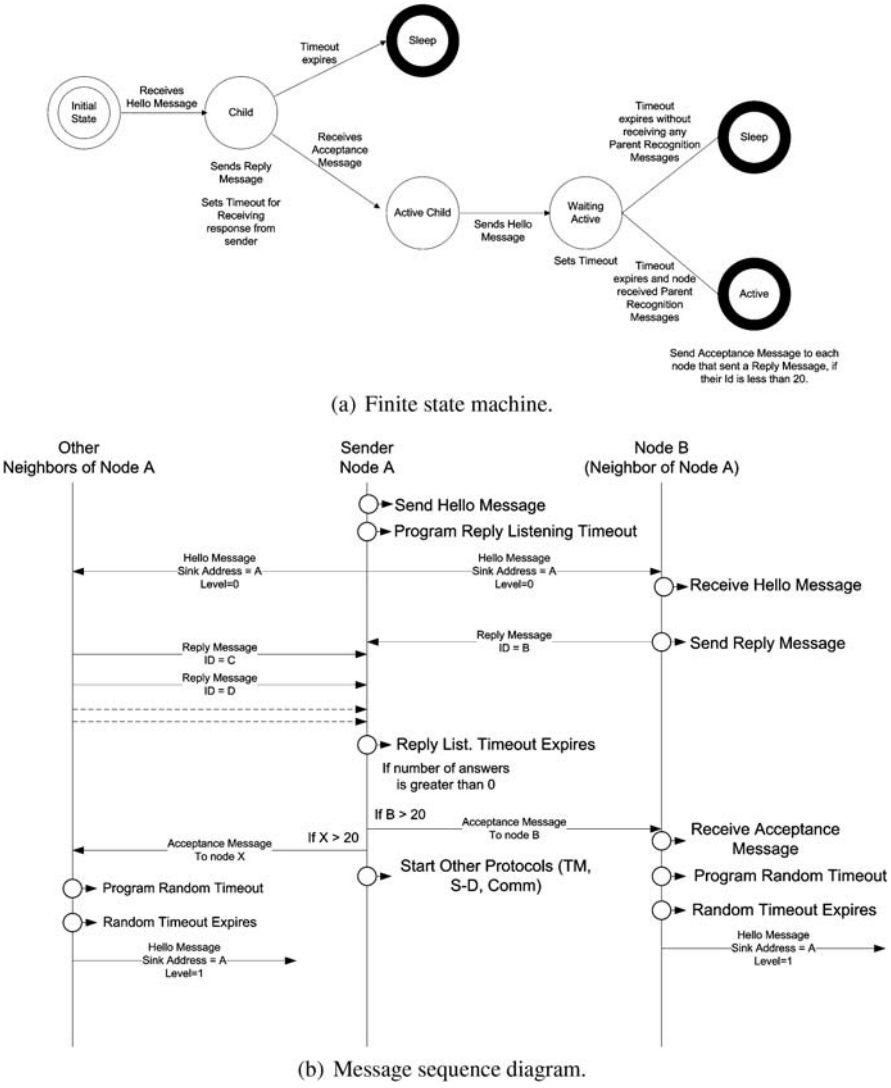


Fig. A.2 Useful diagrams to design a communication protocol.

cases where nodes change several times from one state to another during the execution of the protocol, and by a timeline of message exchanges. For this example protocol, these diagrams are shown in Figure A.2.

The first part of the algorithm is a *HELLO-REPLY* sequence that is used by many protocols in the neighborhood discovery process. The message sequence is simple: one message announces the presence of a node, and a set of nodes, whose size is unknown, answer back with a reply message. Given that the initiator has no idea of how many nodes are within its transmission range, it waits for a certain amount of

time. This timer can be static (a fixed value or a random value defined on the fly) or dynamic (value is changed after the reception of a new reply message).

The second part of the protocol consists of the selection and notification processes. In this case, the sender node selects the next generation of possible active nodes based on a policy (all nodes with ID less than 20), and notifies them one by one using unicast messages.

The third part is the initiation of the other protocols. Remember that topology construction protocols are only used to reduce the size of the initial topology, but they do not necessarily take care of maintaining this topology during its operation, or send data messages to the sink. If the user is at the early stages of the protocol design, running experiments with the topology construction protocol only is very convenient; however, for more complex experiments that include network lifetime measurements it is necessary to start the topology maintenance protocols.

A.4 How to Use Atarraya

Atarraya offers a variety of options for designing and experimenting with topology control algorithms. This section provides an in-depth user guide on how to use the simulation tool and all its available options.

The first step is to have a clear understanding of the simulation scenarios to be run. Here, the user needs to know in advance which protocols he or she wants to use; whether the experiment is just a preliminary test or an exhaustive performance evaluation; what is the nature of the topologies that she is planning to use; what type of statistics are needed, and so forth. In this section, these questions are answered in order for a user to create and run successful simulations with Atarraya.

A.4.1 Selection of the Protocols

Atarraya includes four types of protocols: Topology construction, topology maintenance, sensor-data management, and communication-routing protocols. Atarraya can be set to work in either of the following two modes related to topology control: Topology construction only, or All protocols. The first mode is designed to test a specific topology construction algorithm and measure the initial reduced topology that the algorithm produces. In order to select this mode, the user only needs to select a topology construction protocol.

The second mode is intended to test not only the reduced topology but the lifetime of the network, based on the combination of all the protocols, i.e., topology construction and topology maintenance. In order to select the second mode the user needs to select a protocol in each of the protocol categories, i.e., select a topology construction and a topology maintenance protocol, otherwise Atarraya will not allow the user to run the experiment.

The topology construction protocols included in Atarraya are based on algorithms presented in published papers. Currently, Atarraya includes the algorithms evaluated in this book, i.e., A3, EECDS, and CDS-Rule-K. Although all types of

topology construction protocols might be implemented, such as those based on changing the transmission range of the nodes, hierarchical protocols, cluster-based protocols, etc., the current version of Atarraya includes only those based on the Connected Dominating Set concept. In addition, and for the sake of comparison with a wireless sensor network without topology control, Atarraya offers a protocol that does not reduce the topology at all, called JustTree. The only service that this protocol provides is the creation of a forwarding tree to implement the constant gateway forwarding protocol.

Atarraya also includes all the topology maintenance techniques included in Part II of this book. They are generic algorithms that work with any of the topology construction protocols. The simplest topology maintenance protocol included in Atarraya is the No Topology Maintenance protocol, which does nothing to maintain the initial topology, but monitors it until it dies. The protocol is in charge of informing Atarraya when this event happens. In Atarraya, the termination policy is defined as the moment at which the sink stops receiving information from the nodes.

The **sensor-data management protocol** models the behavior of the sensors, regarding variables like data transmission frequency, data aggregation policies, etc. Atarraya provides a simple protocol for sending and receiving messages without data aggregation. Nodes transmit data packets at predefined times, and forward every received data packet based on the forwarding policy explained in the following paragraph.

Although communication or routing protocols are not the focus of this simulator, some kind of routing procedure is necessary so packets can reach the sink. Given that the topology control protocols implemented in Atarraya are designed to produce a tree-like reduced topology, the tool provides a very simple forwarding algorithm that allows packets to reach the sink node: The **constant gateway forwarding protocol**. In this protocol, packets are always forwarded to a default gateway unless the destination of the packet is a direct neighbor, in which case it will be sent directly to that node. In a tree-like topology, the gateway of a node is simply its parent node. In this fashion, the packet will finally reach the sink node, since it is the root of the tree.

Atarraya does not include any routing protocol, but defines a data structure that models a routing table. This data structure allows users to develop more advanced routing protocols than the one currently implemented. In addition, the routing table can store a limited amount of packet sequence numbers (events have a field for this purpose) that allow the implementation of other forwarding algorithms like flooding-based protocols, or save the last versions of the routing protocol information packets, like routing tables on a Distance Vector protocol, or the last neighborhood information in a Link-State protocol.

In Atarraya's graphical user interface, the panel named *Atarraya* presents the available protocols. In addition, this panel contains the controls for the simulation agent, the report configuration options, simulation events and statistics panels, and the batch simulations controls. This panel is shown in Figure A.3.

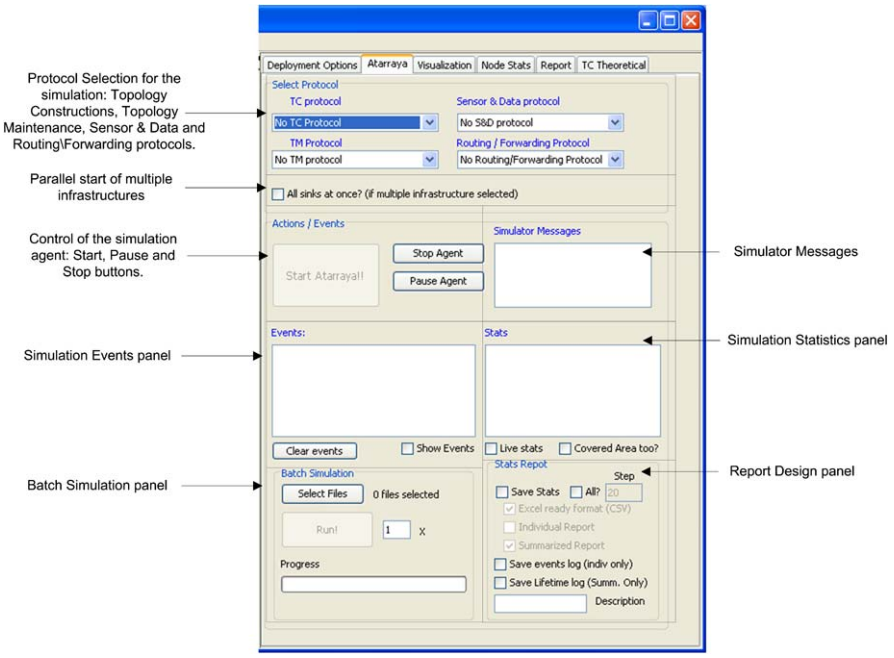


Fig. A.3 Simulation control panel.

A.4.1.1 Other Protocols

In order to write this book, several experiments were run to validate analytically derived equations to show special effects or behaviors of topology control algorithms. This section describes five tools included in Atarraya that are very well suited for educational purposes, as they allow users to:

- Calculate the Critical Transmission Range (CTR) based on the formulas of Penrose–Santi and Santi–Blough.
- Reproduce the experiment to obtain the Giant Component figures: Greatest Connected Component and Ratio of Connected Topologies.
- Reproduce the experiment that proves connectivity of the CTR formula of Santi-Blough for 2 dimensional deployments.
- Calculate the Minimal Spanning Tree on a given graph and provide the sum of the selected edges.
- Save the neighborhoods of a graph in a file.

The first three tools were utilized in Chapter 7. The user can reproduce those experiments with different parameters if so desired. The Minimal Spanning Tree of a graph is still a classical tool for graph analysis – Prim's algorithm was implemented. Regarding the last tool, the information provided by it can be used to define and solve linear programming optimization problems on graphs, like finding a minimal set cover of the graph. The panel that contains all these tools is shown in Figure A.4.

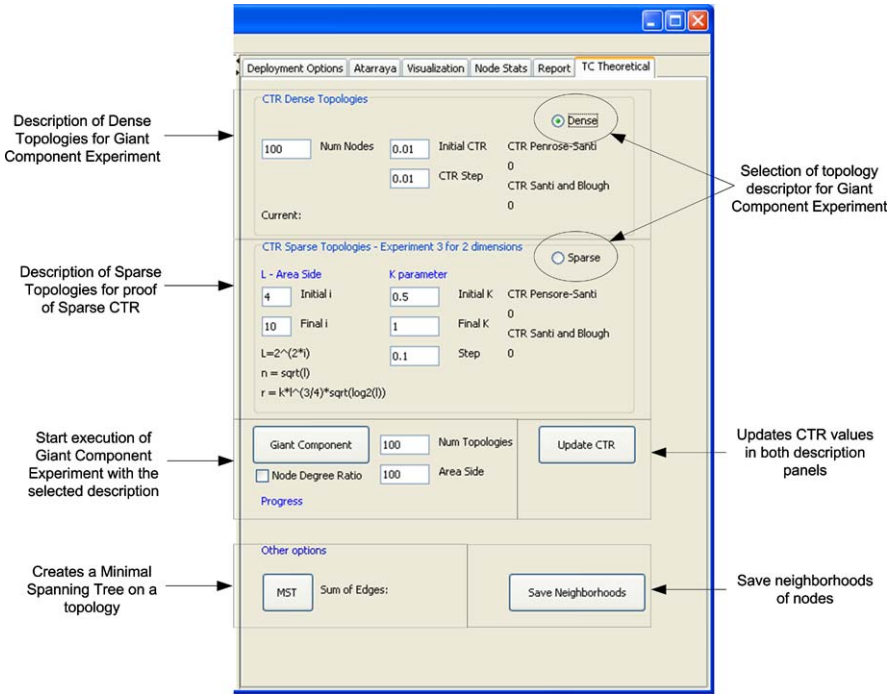


Fig. A.4 Other protocols for educational purposes.

A.4.1.2 Energy and Communications Model

In the design of Atarraya simplicity and focus on reaching a better understanding of the behavior of the topology control algorithms was embraced. As such, several assumptions were made to make the simulator simpler while still good enough for achieving its main objective. These assumptions are related to the energy and the communication models utilized in the tool.

The energy model included in Atarraya is based on the following formulas, taken from [47]:

$$E_{TXbit} = E_{elect} + \left(E_{amp} \cdot (\pi r^2)\right)$$

$$E_{RXbit} = E_{elect}$$

In addition, Atarraya also makes the following assumptions:

- During the idle time, a node does not spend energy. Even though this assumption has been proven untrue because being idle might be as costly as receiving data, this is still an assumption that can be done in most experiments, since the most important factor is the overhead in terms of message exchange and its associated cost.
- The nodes are assumed to have one radio for general messages and a second radio for control messages: The main radio is used in all operations when the

node is in active mode, and the second one (low power cheaper radio) is used to send and receive control packets to "wake up" the main radio. Only the main radio can be turned off, which means no messages will be received and no energy will be used. The secondary radio is assumed to use half the energy of the main radio.
- The sink node has a infinite source of energy. In general, the sink node is assumed to be powered from an external source of energy.

The communication model used in Atarraya is based on the following assumptions:

- The communication range of the nodes is a perfect symmetric unit disk. If $d_{x,y} \leq r_x \rightarrow x$ and y can see each other.
- A constant bit error rate (BER) is defined for the whole network. This is a simple implementation of an error model. Whenever a packet is going to be sent, a random number is generated and compared to the message error rate (that depends on the size of the message). If the random number is greater, the message is received, otherwise it is lost. The default value for the BER is 0, which means there is no packet loss. No sum of partially received packets will build a complete packet.
- Atarraya assumes that there exists a Data Link Layer that deals with packet losses and retransmissions, but it does not model this. In order to model some of the consequences of the operations of the MAC layer, the packets are delayed a random amount of time in order to model delays occurring due to retransmissions, contention, etc. The variable that defines the maximum delay value can be found in the *constants* interface, by the name of *MAX_TX_DELAY_RANDOM*. The default value for this variable is 0.2 time units.

A.4.2 Type of Experiments

Atarraya offers two types of experiments: single topology based, that uses the visual representation of the topology, and the batch execution mode that simulates a large set of topologies. The single topology based type is good for protocol design and debugging. The batch type is made for full scale evaluation and analysis.

During the protocol design phase, it is very important to have the capability to run a small number of controlled topologies one at a time to be able to compare the results of several runs on a known scenario. This is a debugging phase where many changes are introduced in the protocol until it behaves as intended. During this process, a visual representation of the topology and the performance of the topology control algorithm is very helpful.

Once the protocol has reached a stable version that has worked well in several single topologies, the protocol needs a more exhaustive test with a larger number of topologies in order to analyze its average behavior. The batch execution mode allows the researcher to run simulations with hundreds of random topologies. In this case, the visual representation of the topologies is not necessary; actually, it would slow down the simulation process.

In the single topology mode, the topology is loaded using the *Load Topology* item in the *File* menu tab, or the *Deployment Options* tab, which is generated by the user (the generation of topologies will be explained in the following section). In the batch execution mode, several topologies are loaded from files selected by the user. The batch execution controls are located in the *Atarraya* panel, as shown in Figure A.3.

In order to increase the randomization of the simulation process, Atarraya introduces some noise on some common processes in the network, like message transmission delay, so each instance of a simulation, even working on the same topology, would produce different outcomes. Atarraya allows multiple replicas of each topology, which is useful to obtain the average results and the variability of a single topology that shows the confidence of the algorithm.

A.4.3 Structure of a Topology

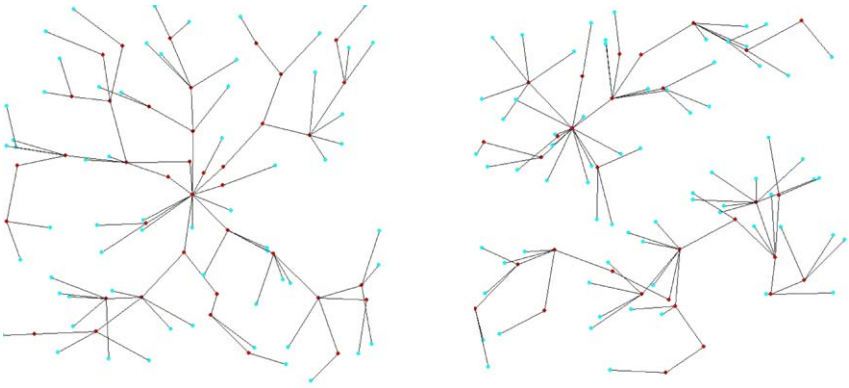
A topology in Atarraya is composed of four basic elements: Deployment area, regular nodes, sink nodes, and virtual network infrastructures or VNI. The deployment area is an abstract concept, which is useful for visualization purposes. It is a rectangle in which the user deploys the nodes of the network. In order to define the deployment area, the user needs to define its width and height.

The set of regular nodes is usually the biggest set of elements in the topology. They are in charge of monitoring the environmental variable or variable of interest, sending this information to the sink and routing packets. As with any wireless sensor devices, regular nodes are very limited in terms of resources.

The sink nodes are special nodes that, in most scenarios, are included to receive the information from all active nodes in the network. They serve as bridges between the wireless sensor network and any type of external network used to transport the sensed data to its final destination somewhere else in the Internet. In some cases they are also in charge of initiating, executing, and/or controlling the topology construction and maintenance protocols, routing protocols, etc.

Despite the fact that in real life the hardware configuration of the sink nodes is different compared to a regular wireless sensor device, Atarraya uses the same data structure to model both nodes, with the main difference that sink nodes are considered to have an unlimited amount of energy.

One contribution of Atarraya is the introduction of the concept of **Virtual Network Infrastructures (VNI)**. This idea is an abstraction used to allow overlaying topologies over the same set of nodes. Imagine having several sink nodes on the initial topology, and that each one builds a reduced topology rooted at itself. Using this abstraction we can rotate not only sink nodes, but also complete topologies. This approach was used to implement the static global topology maintenance techniques defined in Chapter 11. The initial idea was to let the network operate in the same way the lights in a Christmas tree rotate: a certain number of subsets take turns or shifts to be active during a certain amount of time, and go to sleep until the next turn.



(a) Single VNI, single sink. (b) Single VNI, multiple sinks.

Fig. A.5 Different topology designs generated by Atarraya.

Atarraya identifies the different VNIs with numbers from 0 to 6, and visually with colors: Black, Red, Blue, Green, Orange, Pink, and Yellow. Each node is assumed to have a separated data structure for each VNI, so the information of each one is independent from each other. Regular nodes have no affiliation with a particular VNI, while each sink node is associated with a VNI to which it serves as a sink node. All sink nodes are assumed to be regular nodes by the VNI, that are not associated with them, keeping their characteristic of unlimited energy.

The following list contains all the available options to define the network topology:

- Sink or No Sink: Even though most wireless sensor networks contain one or more sink nodes, Atarraya allows the user to define a wireless sensor network without sinks.
- One or multiple sinks in a single VNI: Atarraya allows to define a single network with a single sink or multiple sinks. Figure A.5(a) shows this first case, which facilitates the design because there is only one active element in charge of organizing the protocols. Nonetheless, Atarraya also allows for network designs with multiples sinks in a single network, as shown in Figure A.5(b). Including many sinks distributed in the area of deployment can be a good idea to reduce the average path length; however, having multiple sinks can also cause network partitions, especially if their structures are disjoint.
- One VNI or multiple VNIs: Having multiple VNIs is the way in which the static global topology maintenance techniques were implemented, where there are several subsets of topologies and one sink.

A.4.4 Structure of the Nodes

In terms of the nodes, Atarraya can manage homogeneous networks, in which all nodes have the same characteristics, and heterogeneous networks, where there is at least one node that is different from the others.

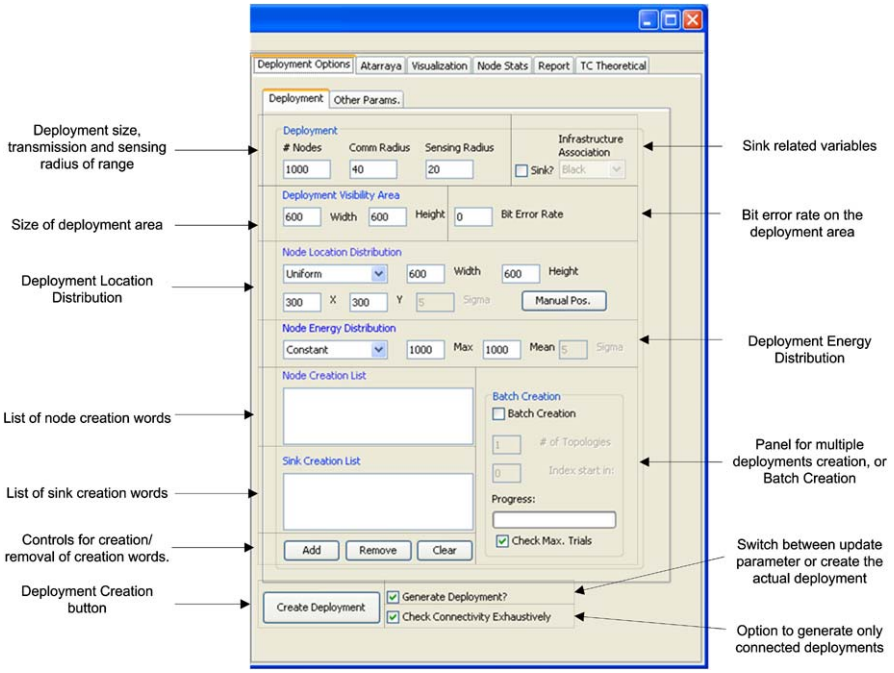


Fig. A.6 Deployment definition panel.

When creating a topology, the simulator works based on subsets of homogeneous nodes. Each set defines a family of nodes that share the same characteristics. For example, the user can define a set of "weak" nodes with low transmission range and low energy, and a set of "powerful" nodes with high transmission range and higher energy. Atarraya also allows for different random distributions for the location and energy of each group of nodes, that will allow the user to create denser zones in the topology, or zones with nodes that have less energy.

The data structure that models these families of nodes is called *Creation Words*. These creation words are created based on the values defined on the *Deployment Options* panel in the simulator (Figure A.6). The user can create as many creation words as desired: these families can model from a single node to the complete set of regular nodes. In the panel there are two list boxes where the creation words are stored until the topology is created: The regular nodes list (top) and the sink creation words (bottom). There are three buttons for adding a new creation word, removing a selected creation word, and clearing both list boxes.

The parameters that can be defined in a homogeneous family of nodes are:

- Number of nodes.
- Communication radius.
- Sensing radius.
- Size of the area of deployment.
- Position distribution:

  - Uniform, with center in (x, y) and the limits defined by the deployment area parameters.
  - Normal with center in (x, y), requiring mean and standard deviation of the location.
  - Grid H-V: cover the deployment area with a grid where nodes are connected of its horizontal and vertical adjacent nodes in the grid. Number of nodes is not a parameter for this option. Distance between the nodes is the communication radius, $comm_{radius}$.
  - Grid H-V-D: cover the deployment area with a grid where nodes are connected of its horizontal, vertical and diagonal adjacent nodes in the grid. The number of nodes is not a parameter for this option either. The distance between the nodes is $comm_{radius}\sqrt{2}$.
  - Constant position at (x, y).
  - Center of area, based on the size of the deployment area parameters.
  - Manual: press the button and click on the panel to select the position of the node, as many times as defined on the number of nodes of the subset.

- Energy distribution:
  - Constant value.
  - Normal, requiring mean and standard deviation of the energy function.
  - Poisson, requiring lambda of the energy distribution.
  - Uniform, requiring maximum of the energy function.
- Inter-query time: Frequency of querying the sensor for readings. This parameter is valid only when a sensor-data protocol is selected.

In the case of the sink node set, two extra parameters must be defined:

- Sink?: select the check box if you want the nodes in the set to be included as sink nodes.
- VNI Selection: select the VNI identifier associated to the sinks of the set.

Atarraya also includes some other special purpose variables, as follows:

- Inter-query time: Time period between reading the sensor and transmitting the data packet to the sink node.
- Inter-reset time: Time period between the time-triggered topology maintenance protocols.
- Energy threshold: Energy percentage differential used to invoke energy-triggered topology maintenance protocols. Every time the energy of a node crosses this energy threshold value, since the last invocation of the topology maintenance protocol, the node will invoke the protocol again. For example, if the battery is fully charged and the energy threshold is set to 0.10, the node will invoke the protocol for the first time when 90% of its energy is consumed.

All these options can be found in the *Deployment Options* tab, under the tabs named *Main Parameters* and *Other Parameters*. The Deployment Options and Deployment's other options panels are shown in Figures A.6 and A.7.

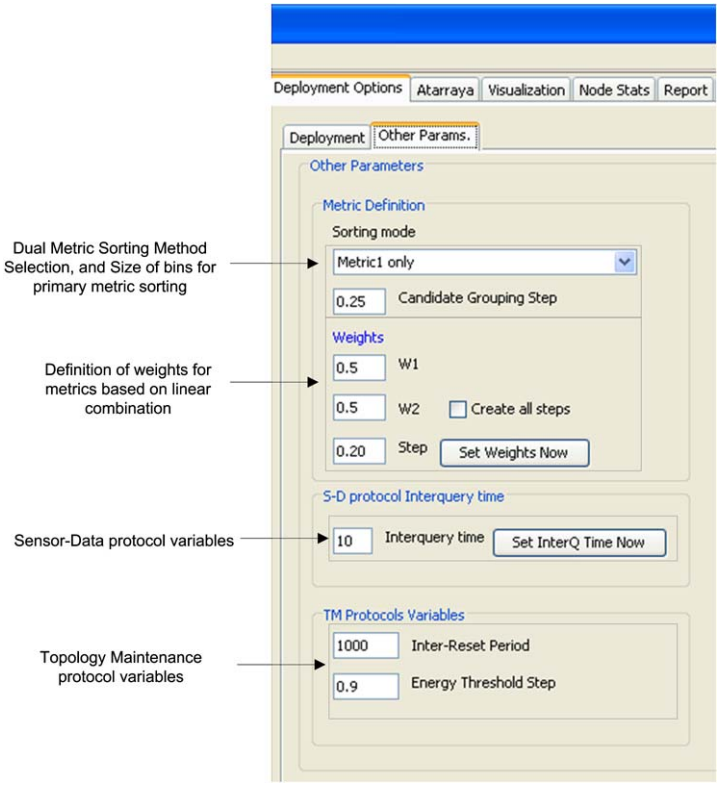


Fig. A.7 Other parameters for deployment definition.

A.4.5 Simulation Results

Atarraya offers three types of result logs that can be obtained from a set of experiments: General statistics, network lifetime, and the simulation event Logs. The **general statistics log** can register the state of several variables in a periodical manner, or provide just one snapshot of the network at the end of the simulation. The most usually consulted variables are:

- Simulation clock.
- Number of nodes.
- Number of sink nodes.
- Number of active nodes.
- Number of dead nodes.
- Average node degree.
- Average level of nodes (if level is used by the protocol).
- Number of messages sent and received.
- Number of data messages received by the sink.
- Energy on the tree, energy spent, etc.
- Area of communication coverage.

```
Clock: 36.905478263665884
# of Nodes: 100
# of Sinks: 1
Not Covered: 0
Ratio: 0.0
Not Visited: 0
Avg. Level: 3.603960396039604
Avg. Num. Neighb.: 7.762376237623762
Reachable. Num. Active Neighb. from Sink: 26
# of Messages regular sent: 317
# of Messages long sent: 26
# of Messages regular received: 1697
# of Messages long received: 136
# of Lost Messages regular: 0
# of Lost Messages long: 0
# of Data Messages received by sink: 0
# of dead nodes: 0
Real Parents in Tree 0 =26
Real Parents in Tree 1 =0
Real Parents in Tree 2 =0
Real Parents in Tree 3 =0
Real Parents in Tree 4 =0
Total Energy initial=101000.0
Total Energy final=100947.62194912211
Total Energy spent=53.07221014322563
Total Energy spent ratio=5.185945631473989E-4
Total Energy in tree=25979.027257565518
Ratio Energy=0.25735155277515126
Covered Area=0.907875
Error in simulation=no error
```

Fig. A.8 Example of statistics generated by Atarraya.

These values are stored in a text file where each individual row contains a snapshot of the variables of the network in the moment at which data was collected. If a experiment includes more than one topology, the user can decide if the data is going be stored individually per execution, or if all the results are going to be summarized in a single file. The usual format in which data is presented is in comma separated values (.csv), which is readable by most data analysis programs, like Excel, Matlab, etc. However, if the user does not want the results to be saved in files, the *Report* panel has a text area that holds the statistics generated by the experiments in csv format. If just one topology is simulated, the user can use the *Stats* text area in the *Atarraya* panel, which presents the results in plain text format. Figure A.8 shows one example of the simulation results generated by Atarraya.

The **network lifetime log** registers the status of the active topology. This log stores information every time the topology maintenance protocol is invoked, or when a node dies. This specific log cannot be stored in individual files per execution; it is stored in a summarized format in csv format.

The information stored in the network lifetime log by a single topology is represented in four rows. The first row registers the moment in which the information of the network was calculated. The second row represents the number of active nodes

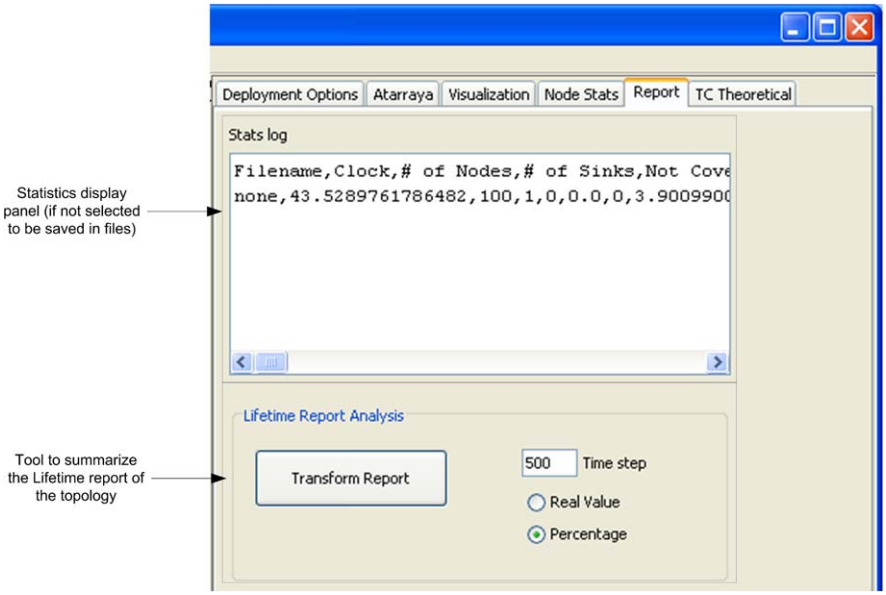


Fig. A.9 Report panel.

that can still reach the sink node – those that still can provide information to the sink. This value is important because if the sink gets isolated, no matter how many active nodes remain, all of them are useless because the information they produce gets lost. In the case of having more that one VNIs, the program works with the active VNI in the nodes. The third row shows the ratio between the number of active nodes that can reach the sink (value found on the second row) and the maximum number of active nodes. This value is the percentage of active nodes that are still alive and connected to the sink. Finally, the fourth row contains the percentage of covered area by the active nodes in the second row. This information can be very useful in order to compare efficiency between protocols, in terms of number of active nodes and real covered area.

Atarraya offers a way to summarize this log by calculating the average number of active nodes for the complete set of executions that are stored in a single file. The idea is to create a single lifetime function that represents the average number of active nodes and wraps all the individual lifetimes. Atarraya creates a discrete time line, in which the size of each cell is based on the parameter "Time step". Each cell contains the global average of all experiments during that specific amount of time. The *Report* panel presents the controls for this operation, as shown in Figure A.9.

The **simulation event log** registers all the events generated by the simulator during the execution of a single topology. The complete set of events allows the user to debug the protocol by seeing the sequence of operations executed. This information is only available in individual files per topology.

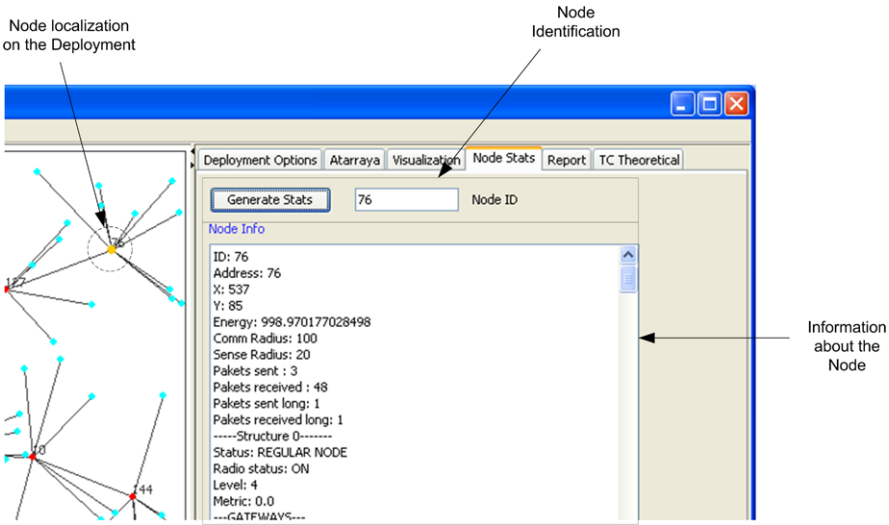


Fig. A.10 Node statistics panel.

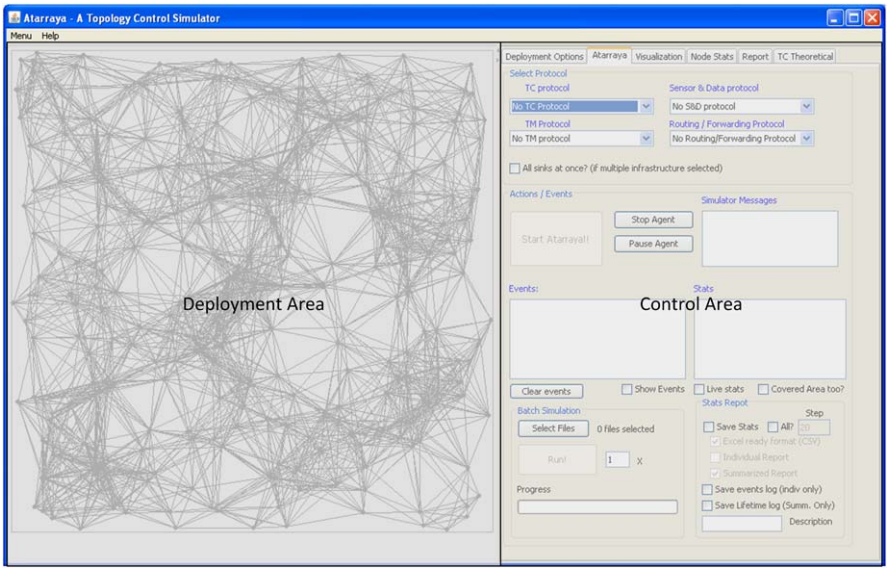


Fig. A.11 Main window description.

Atarraya not only offers statistics from the complete simulation point of view, but from the individual node perspective. It is always very useful to know the status of the nodes at any given point in time to check if the algorithms are working as desired. This information can be found in the *Node Stats* panel, as shown in Figure A.10. In order to get the information of a node, the user can do it in two ways:

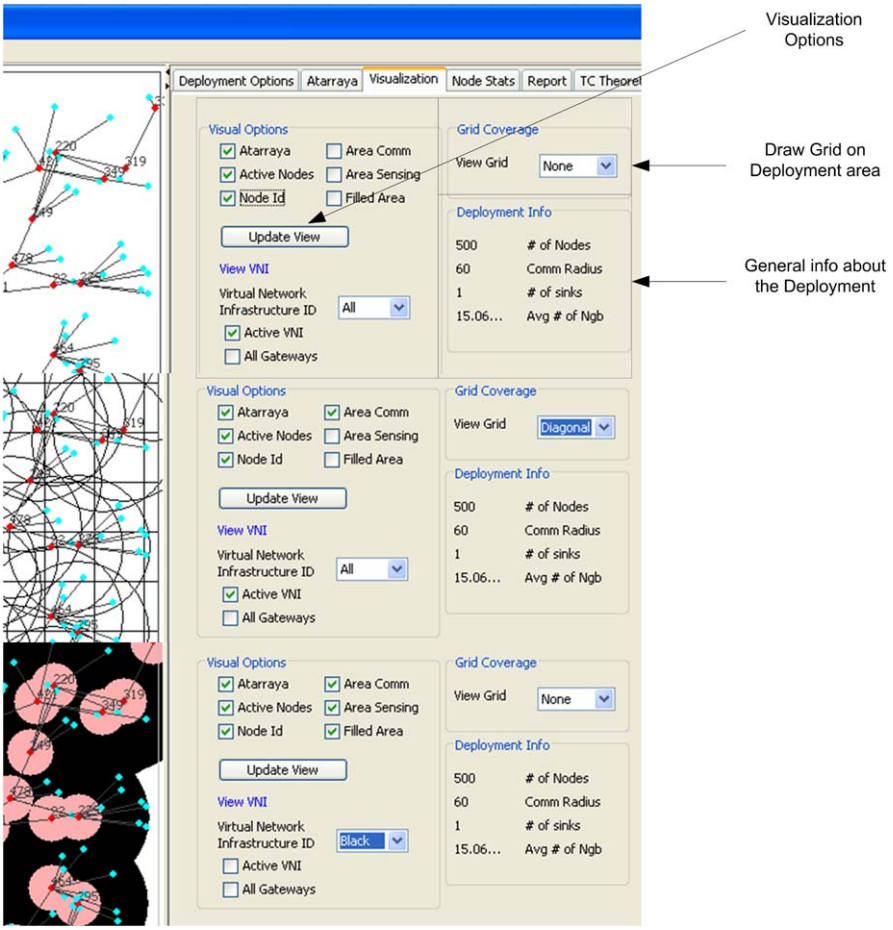


Fig. A.12 Visualization control panel.

dragging the mouse pointer to the desired node and clicking over it, or typing the id number of the node in the text field and press the button Generate Stats. Once the node is selected, the point in the topology turns orange and increases in size.

Atarraya also allows for the visual representation of the reduced topology. The main window of Atarraya is divided in two environments: The deployment visualization area (left) and the control area (right), as shown in Figure A.11. One of the panels in the control area contains the visualization options: changing view from MaxPower graph to the reduced topology (Atarraya mode); showing the active nodes (Parent mode), communication and sensing coverage areas, and node id's; drawing a grid over the deployment the area, etc. The combination of some of these options are shown in Figure A.12, where on the left side of the figure the deployment visualization area with the different presentation options can be seen.

A.5 Future of Atarraya

Atarraya is a very useful tool for testing topology control algorithms, but it is still far from reaching its full potential. There is a lot of room for future improvements. For example, transmission range control topology construction protocols, local topology maintenance protocols, data aggregation algorithms, routing and forwarding protocols, 3D scenarios, etc. could be included. More complex and realistic sensing and communication models and data link layer protocols could be incorporated as well.

It is hoped that making Atarraya freely available the research community will contribute to its expansion. Maintaining Atarraya's Website with the most recent versions of the tool, its upgrades and features, for the benefit of the entire community and the research area is a major commitment. The main Atarraya Website is at http://www.csee.usf.edu/~labrador/Atarraya. Please contact the authors with your own upgrades, suggestions, corrections, etc.

References

1. Abramson, N.: The ALOHA system: another alternative for computer communications. In: Proceedings of AFIPS, pp. 281–285 (1970)
2. Akyildiz, I.F., Melodia, T., Chowdhury, K.R.: A survey on wireless multimedia sensor networks. Comput. Netw. **51**, 921–960 (2007)
3. Alzoubi, K.M., Wang, P.J., Frieder, O.: Distributed heuristics for connected dominating set in wireless ad hoc networks. Commun. Netw. **4**(1), 22–29 (2002)
4. ATMEL: ATMega 128L microcontroller data sheet. http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf
5. Bao, L., Garcia-Luna-Aceves, J.J.: A new approach to channel access scheduling for ad hoc networks. In: Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, pp. 210–221 (2001)
6. Bao, L., Garcia-Luna-Aceves, J.J.: Topology management in ad hoc networks. In: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 129–140 (2003)
7. Basagni, S., Chlamtac, I., Syrotiuk, V.R., Woodward, B.A.: A distance routing effect algorithm for mobility (DREAM). In: Proceedings of Fourth ACM/IEEE Mobicom Conference, pp. 76–84 (1998)
8. Baum, L.E., Petrie, T., Soules, G., Weiss, N.: A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann. Math. Stat. **41**(1), 164–171 (1970)
9. Bharghavan, V., Demers, A., Shenker, S., Zhang, L.: MACAW: a media access protocol for wireless LAN's. In: Proceedings of ACM SIGCOMM, pp. 212–225 (1994)
10. Blough, D.M., Leoncini, M., Resta, G., Santi, P.: K-Neighlev: a practical realization of neighbor-based topology control in ad hoc networks. Technical Report IIT-TR-09/2003, Istituto di Informatica e Telematica (2003)
11. Blough, D.M., Leoncini, M., Resta, G., Santi, P.: The K-Neigh protocol for symmetric topology control in ad hoc networks. In: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 141–152 (2003)
12. Borbash, S.A., Jennings, E.H.: Distributed topology control algorithm for multihop wireless networks. In: Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 355–360 (2002)
13. Bose, P., Morin, P., Stojmenovic, I., Urrutia, J.: Routing with guaranteed delivery in ad hoc wireless networks. In: Proceedings of the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 48–55 (1999)
14. Bose, P., Morin, P., Stojmenovic, I., Urrutia, J.: Routing with guaranteed delivery in ad hoc wireless networks. Wirel. Netw. **7**(6), 609–616 (2001)

15. Braginsky, D., Estrin, D.: Rumor routing algorithm for sensor networks. In: Proceedings of First ACM International Workshop on Wireless Sensor Networks and Applications, pp. 22–31 (2002)
16. Broch, J., Johnson, D., Maltz, D.: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. Addison–Wesley, Reading (2001)
17. Busse, M., Haenselmann, T., Effelsberg, W.: TECA: a topology and energy control algorithm for wireless sensor networks. In: Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, pp. 317–321 (2006)
18. Cartigny, J., Ingelrest, F., Simplot-Ryl, D., Stojmenovic, I.: Localized LMST and RNG based minimum-energy broadcast protocols in ad hoc networks. Ad Hoc Netw. **3**(1), 1–16 (2005)
19. Cerpa, A., Estrin, D.: ASCENT: adaptive self-configuring sensor networks topologies. IEEE Trans. Mobile Comput. **3**(3), 272–285 (2004)
20. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. Wirel. Netw. **8**(5), 481–494 (2002)
21. Chessa, S., Santi, P.: Comparison-based system-level fault diagnosis in ad hoc networks. In: Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems, pp. 257–266 (2001)
22. Chew, P.: http://www.cs.cornell.edu/Info/People/chew/Delaunay.html
23. Cisco: http://www.cisco.com/en/US/products/hw/wireless/index.html
24. Clementi, A., Penna, P., Silvestri, R.: Hardness results for the power range assignment problem in packet radio networks. In: Proceedings of 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems: Randomization, Approximation, and Combinatorial Algorithms and Techniques, pp. 197–208 (1999)
25. Crossbow: http://www.xbow.com/
26. van Dam, T., Langendoen, K.: An adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, pp. 171–180 (2003)
27. Das, B., Bharghavan, V.: Routing in ad-hoc networks using minimum connected dominating sets. In: Proceedings of IEEE International Conference on Communications, vol. 1, pp. 376–380 (1997)
28. Das, B., Sivakumar, E., Bharghavan, V.: Routing in ad-hoc networks using a virtual backbone. In: Proceedings of International Conference on Computer Communications and Networks (1997)
29. Deb, B., Bhatnagar, S., Nath, B.: Information assurance in sensor networks. In: Proceedings of the 2nd ACM Workshop on Wireless Sensor Networks (2003)
30. Deb, B., Bhatnagar, S., Nath, B.: ReInForM: reliable information forwarding using multiple paths in sensor networks. In: Proceedings of IEEE LCN, pp. 406–414 (2003)
31. Dette, H., Henze, N.: The limit distribution of the largest nearest neighbor link in the unit D-cube. J. Appl. Probab. **26**, 67–80 (1997)
32. Dunkels, A., Voigt, T., Alonso, J.: Making TCP viable for wireless sensor networks. In: Proceedings of European Workshop on Wireless Sensor Networks (2004)
33. El-Hoiydi, A.: Spatial TDMA and CSMA with preamble sampling for low power ad hoc wireless sensor networks. In: Proceedings of the IEEE International Conference on Computers and Communications, pp. 685–692 (2002)
34. El-Hoiydi, A., Decotignie, J.: WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks. In: Proceedings of International Symposium on Computers and Communications, pp. 244–251 (2004)
35. Elhadef, M., Boukerche, A., Elkadiki, H.: Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols. In: Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access, pp. 18–27 (2006)
36. Frye, L., Cheng, L., Du, S., Bigrigg, M.: Topology maintenance of wireless sensor networks in node failure-prone environments. In: Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, pp. 308–312 (2006)

37. Fullmer, C.L., Garcia-Luna-Aceves, J.J.: Floor acquisition multiple access (FAMA) for packet-radio networks. In: Proceedings of ACM SIGCOMM, pp. 262–273 (1995)
38. Gabriel, K.R., Sokal, R.R.: A new statistical approach to geographic variation analysis. Syst. Zool. **18**, 259–270 (1969)
39. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
40. Gelal, E., Jakllari, G., Young, N., Krishnamurthy, S.V.: An integrated scheme for fully-directional neighbor discovery and topology management in mobile ad hoc networks. In: Proceedings of IEEE International Conference on Mobile Ad Hoc and Sensor Systems, pp. 139–149 (2006)
41. Gilbert, E.N.: Capacity of a burst-noise channel. Bell Syst. Tech. J. **39**, 1253–1266 (1960)
42. Godfrey, P.B., Ratajczak, D.: Naps: scalable, robust topology management in wireless ad hoc networks. In: Proceedings of the Third International Symposium on Information Processing in Sensor Networks, pp. 443–451 (2004)
43. Goodman, D.J., Valenzuela, R.A., Gayliard, K.T., Ramamurthy, B.: Packet reservation multiple access for local wireless communications. IEEE Trans. Commun. **37**(8), 885–890 (1989)
44. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. In: Proceedings of European Symposium on Algorithms, pp. 179–193 (1996)
45. Guo, C., Zhong, L.C., Rabaey, J.M.: Low power distributed MAC for ad hoc sensor radio networks. In: Proceedings of IEEE Globecom, pp. 2944–2948 (2001)
46. Haas, Z.J., Halpern, J.Y., Li, L.: Gossip-based ad hoc routing. IEEE/ACM Trans. Netw. **14**(3), 479–491 (2006)
47. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd International Conference on System Sciences (HICSS), pp. 1–10 (2000)
48. Hendrick, C.: Routing information protocol. RFC 1058, IETF (1988)
49. Holger, K., Willig, A.: Protocols and Architectures for Wireless Sensor Networks. Wiley, New York (2005)
50. Holst, L.: On multiple covering of a circle with random arcs. Appl. Probab. **17**, 284–290 (1998)
51. Hou, T., Li, V.: Transmission range control in multihop packet radio networks. IEEE Trans. Commun. **34**(1), 38–44 (1986)
52. IEEE: IEEE 802.11-2007. IEEE standard for information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. http://standards.ieee.org/getieee802/802.11.html
53. IEEE: IEEE 802.15 WPAN Task Group 4 (TG4). http://www.ieee802.org/15/pub/TG4.html
54. IEEE: IEEE 802.15.1-2005. IEEE standard for information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs). http://standards.ieee.org/getieee802/802.15.html
55. IEEE: IEEE 802.3-2005. IEEE standard for information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. http://standards.ieee.org/getieee802/802.3.html
56. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: a scalable and robust communication paradigm for sensor networks. In: Proceedings of Sixth Annual International Conference on Mobile Computing and Networking, pp. 56–67 (2000)
57. Intel: StrongARM SA-1100 microprocessor brief data sheet. http://www rfm.com/
58. Iyengar, R., Kar, K., Banerjee, S.: Low-coordination topologies for redundancy in sensor networks. In: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 332–342 (2005)

59. Iyer, Y.G., Gandham, S., Venkatesan, S.: STCP: a generic transport layer protocol for wireless sensor networks. In: Proceedings of IEEE ICCCN Conference (2005)
60. Karger, D., Klein, P., Tarjan, R.: A randomized linear-time algorithm to find minimum spanning trees. J. ACM **42**(2), 321–328 (1995)
61. Karn, P.: A new channel access method for packet radio. In: Proceedings of the ARRL/CEEL Amateur Radio 9th Computer Networking Conference, pp. 134–140 (1990)
62. Karp, B., Kung, H.T.: Greedy perimeter stateless routing for wireless networks. In: Proceedings of ACM/IEEE Mobicom, pp. 243–254 (2000)
63. Kawadia, V., Kumar, P.: Power control and clustering in ad hoc networks. In: Proceedings of INFOCOM, pp. 459–469 (2003)
64. Kirousis, L., Kranakis, E., Krizanc, D., Pele, A.: Power consumption in packet radio networks. Theor. Comput. Sci. **243**, 289–305 (2000)
65. Kleinrock, L., Silvester, J.A.: Optimum transmission radii for packet radio networks or why six is a magic number. In: Proceedings of the IEEE National Telecommunication Conference, pp. 4.3.1–4.3.5 (1978)
66. Kulik, J., Rabiner, W., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: Proceedings of Fifth ACM/IEEE Mobicom Conference, pp. 174–185 (1999)
67. Kumar, V., Arunan, T., Balakrishnan, N.: E-SPAN: enhanced-SPAN with directional antenna. In: Proceedings of IEEE Conference on Convergent Technologies for Asia-Pacific Region, vol. 2, pp. 675–679 (2002)
68. Lettieri, P., Schurgers, C., Srivastava, M.: Adaptive link layer strategies for energy efficient wireless networking. Wirel. Netw. **5**(5), 1022–1038 (1999)
69. Li, L., Halpern, J., Bahl, P., Wang, Y.M., Wattenhofer, R.: Analysis of distributed topology control algorithm for wireless multi-hop networks. In: Proceedings of ACM Symposium on Principles of Distributed Computing, pp. 264–273 (2001)
70. Li, L., Halpern, J.Y., Bahl, P., Wang, Y.M., Wattenhofer, R.: A cone-based distributed topology-control algorithm for wireless multi-hop networks. IEEE/ACM Trans. Netw. **13**(1), 147–159 (2005)
71. Li, N., Hou, J.: Topology control in heterogeneous wireless networks: problems and solutions. In: Proceedings of IEEE INFOCOM (2004)
72. Li, N., Hou, J.C.: FLSS: a fault-tolerant topology control algorithm for wireless networks. In: Proceedings of ACM Mobicom, pp. 275–286 (2004)
73. Li, N., Hou, J.C., Sha, L.: Design and analysis of an MST-based topology control algorithm. In: Proceedings of IEEE INFOCOM, pp. 1702–1712 (2003)
74. Li, X., Song, W., Wang, Y.: Localized topology control for heterogeneous wireless sensor networks. ACM Trans. Sens. Netw. **2**(1), 129–153 (2006)
75. Li, X.Y., Calinescu, G., Wan, P.J.: Distributed construction of a planar spanner and routing for ad hoc wireless networks. In: Proceedings of IEEE INFOCOM, pp. 1268–1277 (2002)
76. Lukachan, G., Labrador, M.A.: SELAR: scalable energy-efficient location aided routing protocol for wireless sensor networks. In: Proceedings of IEEE LCN Workshop on Wireless Local Networks, pp. 694–695 (2004)
77. Lukachan, G., Labrador, M.A.: Scalable and energy-efficient routing protocol for large-scale wireless sensor networks. In: Proceedings of the Sixth IEEE International Caribbean Conference on Devices, Circuits and Systems (2006)
78. Luo, H., Ye, F., Cheng, J., Lu, S., Zhang, L.: TTDD: a two-tier dissemination model for large-scale wireless sensor networks. In: Proceedings of First ACM International Workshop on Wireless Sensor Networks and Applications (2003)
79. Luo, X., Zheng, K., Pan, Y., Wu, Z.: A TCP/IP implementation for wireless sensor networks. In: Proceedings of IEEE Conference on Systems, Man and Cybernetics, pp. 6081–6086 (2004)
80. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless sensor networks for habitat monitoring. In: Proceedings of First ACM Workshop on Wireless Sensor Network Applications (2002)

81. Morris, J.M.: Optimal blocklengths for ARQ error control schemes. IEEE Trans. Commun. **27**(2), 488–493 (1979)
82. Moy, J.: Open shortest path first protocol. RFC 2328, IETF (1998)
83. Murthy, S., Garcia-Luna-Aceves, J.J.: A routing protocol for packet radio networks. In: Proceedings of ACM International Conference on Mobile Computing and Networking, pp. 86–95 (1995)
84. Narayanaswamy, S., Kawadia, V., Sreenivas, R., Kumar, P.: Power control in ad hoc networks: theory, architecture, algorithm and implementation of the COWPOW protocol. In: Proceedings of the European Wireless Conference, pp. 156–162 (2002)
85. Ni, J., Chandler, S.: Connectivity properties of a random radio network. IEE Proc. Commun. **141**(4), 289–296 (1994)
86. Parikh, S., Vokkarane, V., Xing, L., Kasilingam, D.: An energy efficient routing mechanism for wireless sensor networks. In: Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA), vol. 2, pp. 308–312 (2006)
87. Park, V.D., Corson, M.S.: A highly adaptive distributed routing algorithm for mobile wireless networks. In: Proceedings of IEEE INFOCOM, pp. 1405–1413 (1997)
88. Parthasarathy, S., Gandhi, R.: Fast distributed well connected dominating sets for ad hoc networks. Technical Report CS-TR-4559, University of Maryland (2004). http://citeseer.ist.psu.edu/parthasarathy04fast.html
89. Patwari, N., Ash, J., Kyperountas, S., Hero, A., Moses, R., Correal, N.: Locating the nodes. IEEE Signal Process. Mag. **July**, 54–69 (2005)
90. Penrose, M.: The longest edge of a random minimal spanning tree. Ann. Appl. Probab. **7**(2), 340–361 (1997)
91. Penrose, M.: Extremes for the minimal spanning tree on normally distributed points. Adv. Appl. Probab. **30**, 628–639 (1998)
92. Penrose, M.: A strong law for the largest nearest-neighbour link between random points. J. Lond. Math. Soc. **60**(2), 951–960 (1999)
93. Penrose, M.: On K-connectivity for a geometric random graph. Random Struct. Algorithms **15**(2), 145–164 (1999)
94. Penrose, M.: Random Geometric Graphs. Oxford University Press, London (2003)
95. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing. RFC 3561, IETF (2003)
96. Perkins, C.E., Bhagwat, P.: Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of the ACM SIGCOMM Symposium on Communication, Architecture and Protocols, pp. 234–244 (1994)
97. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: Proceedings of Second ACM Conference on Embedded Networked Sensor Systems (2004)
98. Pottie, G.J., Kaiser, W.J.: Wireless integrated sensor networks. Commun. ACM **43**(5), 51–58 (2000)
99. Prim, R.: Shortest connection networks and some generalizations. Bell Syst. Tech. J. **36**, 1389–1401 (1957)
100. Rajendran, V., Obraczka, K., Garcia-Luna-Aceves, J.J.: Energy-efficient, collision-free medium access control for wireless sensor networks. Wirel. Netw. **12**(1), 63–78 (2006)
101. Rappaport, T.S.: Wireless Communications: Principles and Practice. Prentice Hall, New York (2002)
102. Rodoplu, V., Meng, T.H.: Minimum energy mobile wireless networks. IEEE J. Sel. Areas Commun. **17**(8), 1333–1344 (1999)
103. Sankarasubramaniam, Y., Akan, O., Akyildiz, I.F.: ESRT: event-to-sink reliable transport in wireless sensor networks. In: Proceedings of ACM MobiHoc, pp. 177–188 (2003)
104. Sankarasubramaniam, Y., Akyildiz, I.F., McLaughlin, S.W.: Energy efficiency based packet size optimization in wireless sensor networks. In: Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, pp. 1–8 (2003)
105. Santi, P.: Topology Control in Wireless Ad Hoc and Sensor Networks. Wiley, New York (2005)

106. Santi, P., Blough, D.: The critical transmitting range for connectivity in sparse wireless ad hoc networks. IEEE Trans. Mobile Comput. **2**(1), 25–39 (2003)
107. Sayed, A.H., Tarighat, A., Khajehnouri, N.: Network-based wireless localization. IEEE Signal Process. Mag. **July**, 24–40 (2005)
108. Schurgers, C., Tsiatsis, V., Ganeriwal, S., Srivastava, M.: Optimizing sensor networks in the energy–latency–density design space. IEEE Trans. Mobile Comput. **1**(1), 70–80 (2002)
109. Sen, A., Melesinska, E.: On approximation algorithms for radio network scheduling. In: Proceedings of the 35th Allerton Conference on Communication, Control and Computing, pp. 573–582 (1997)
110. Shah, R., Rabaey, J.: Energy aware routing for low energy ad hoc sensor networks. In: Proceedings of IEEE Wireless Communications and Networking Conference (WCNC) (2002)
111. Shakespeare: http://www.shakespeare-military.com/
112. Singh, S., Raghavendra, C.S.: PAMAS – power aware multi-access protocol with signaling for ad hoc networks. ACM Comput. Commun. Rev. **28**(3), 5–26 (1998)
113. Singh, S., Woo, M., Raghavendra, C.S.: Power-aware routing in mobile ad hoc networks. In: Proceedings of ACM International Conference on Mobile Computing and Networking, pp. 181–190 (1998)
114. Sohrabi, K., Gao, J., Ailawadhi, V., Pottie, G.J.: Protocols for self-organization of a wireless sensor network. IEEE Pers. Commun. **7**(5), 16–27 (2000)
115. SPAWAR: http://www.spawar.navy.mil/robots/
116. Stallings, W.: Data and Computer Communications, 7th edn. Prentice Hall, New York (2004)
117. Stann, F., Heidemann, J.: RMST: reliable data transport in sensor networks. In: Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, pp. 102–112 (2003)
118. Stojmenovic, I., Lin, X.: Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. IEEE Trans. Parallel Distributed Syst. **12**(10), 1023–1032 (2001)
119. Stojmenovic, I., Seddigh, M., Zunic, J.: Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks. In: Proceedings of IEEE Hawaii International Conference on System Sciences (2001)
120. Sun, G., Chen, J., Guo, W., Liu, K.J.R.: Signal processing techniques in network-aided positioning. IEEE Signal Process. Mag. **July**, 12–23 (2005)
121. Toussaint, G.: The relative neighborhood graph of a finite planar set. Pattern Recognit. **12**, 261–268 (1980)
122. Tranter, W.H., Shanmugan, K.S., Rappaport, T.S., Kosbar, K.L.: Principles of Communication Systems Simulation with Wireless Applications. Prentice Hall, New York (2004)
123. Wan, C.Y., Campbell, A., Krishnamurthy, L.: PSFQ: a reliable transport protocol for wireless sensor networks. In: Proceedings of 1st ACM International Workshop on Wireless Sensor Networks and Applications, pp. 1–11 (2002)
124. Wan, C.Y., Eisenman, S.B., Campbell, A.T.: CODA: congestion detection and avoidance in sensor networks. In: Proceedings of the ACM Conference on Embedded Networked Sensor Systems (2003)
125. Wan, P.J., Alzoubi, K., Frieder, O.: Distributed well connected dominating set in wireless ad hoc networks. In: Proceedings of IEEE INFOCOM (2002)
126. Wang, Y., Li, X.: Geometric spanners for wireless ad hoc networks. In: Proceedings of 22nd International Conference on Distributed Computing Systems, pp. 171–178 (2002)
127. Wattenhofer, R., Zollinger, A.: XTC: a practical topology control algorithm for ad-hoc networks. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium (2004)
128. Wightman, P., Labrador, M.A.: A3: a topology control algorithm for wireless sensor networks. In: Proceedings of IEEE Globecom (2008)
129. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys), pp. 14–27 (2003)

130. Wu, J., Cardei, M., Dai, F., Yang, S.: Extended dominating set and its applications in ad hoc networks using cooperative communication. IEEE Trans. Parallel Distributed Syst. **17**(8), 851–864 (2006)
131. Wu, J., Dai, F.: An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. IEEE Trans. Parallel Distributed Syst. **15**(10), 908–920 (2004)
132. Wu, J., Dai, F.: Virtual backbone construction in MANETs using adjustable transmission ranges. IEEE Trans. Mobile Comput. **5**(9), 1188–1200 (2006)
133. Wu, J., Li, H.: On calculating connected dominating set for efficient routing in ad hoc wireless networks. In: Proceedings of the 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 7–14 (1999)
134. Xing, G., Lu, C., Zhang, Y., Huang, Q., Pless, R.: Minimum power configuration for wireless communication in sensor networks. ACM Trans. Sens. Netw. **3**(2) (2007)
135. Xu, Y., Bien, S., Mori, Y., Heidemann, J., Estrin, D.: Topology control protocols to conserve energy in wireless sensor networks. CENS Technical Report 0006, University of California (2003)
136. Xu, Y., Heidemann, J., Estrin, D.: Adaptive energy-conserving routing for multihop ad hoc networks. Research Report 527, USC/Information Sciences Institute (2000). http://www.isi.edu/~johnh/PAPERS/Xu00a.html
137. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed energy conservation for ad hoc routing. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, pp. 70–84 (2001)
138. Xue, F., Kumar, P.R.: The number of neighbors needed for connectivity of wireless networks. Wirel. Netw. **10**(2), 169–181 (2004)
139. Yao, A.C.: On constructing minimum spanning trees in K-dimensional spaces and related problems. J. Comput. Soc. Ind. Appl. Math. **11**(4), 721–736 (1982)
140. Ye, W., Heidemann, J., Estrin, D.: An energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of INFOCOM, pp. 1567–1676 (2002)
141. Ye, W., Heidemann, J., Estrin, D.: Medium access control with coordinated, adaptive sleeping for wireless sensor networks. IEEE/ACM Trans. Netw. **12**(3), 493–506 (2004)
142. Younis, O., Fahmy, S.: HEED: a hybryd, energy-efficient, distributed clustering approach for ad hoc sensor networks. IEEE Trans. Mobile Comput. **3**(4), 366–379 (2004)
143. Yuanyuan, Z., Jia, X., Yanxiang, H.: Energy efficient distributed connected dominating sets construction in wireless sensor networks. In: Proceeding of the 2006 ACM International Conference on Communications and Mobile Computing, pp. 797–802 (2006)
144. Zhang, R.: Sink localization and topology control in large scale heterogeneous wireless sensor networks. Ph.D. Dissertation, University of South Florida (2007)
145. Zhang, R., Labrador, M.A.: Energy-aware topology control in heterogeneous wireless multihop networks. In: Proceedings of 2nd IEEE International Symposium on Wireless Pervasive Computing (2007)
146. Zhang, R., Zhao, H., Labrador, M.A.: A scalable and energy efficient sink location service for large-scale wireless sensor networks. Ad Hoc Sens. Wirel. Netw. J. **4**(4), 289–320 (2007)
147. Zou, Y., Chakrabarty, K.: Sensor deployment and target utilization based on virtual forces. In: Proceedings of IEEE INFOCOM (2003)
148. Zou, Y., Chakrabarty, K.: Fault-tolerant self-organization in sensor networks. In: Proceedings of DCOSS, pp. 191–205 (2005)

Index