Winter 2005

Energy-Efficient Self-Organization Protocols for Sensor Networks

Qingwen Xu
Old Dominion University

ENERGY-EFFICIENT SELF-ORGANIZATION PROTOCOLS FOR SENSOR NETWORKS

by

Qingwen Xu
B.S. June 1991, WuHan University
M.S. June 1999, Wake Forest University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2005

ABSTRACT

ENERGY-EFFICIENT SELF-ORGANIZATION PROTOCOLS FOR SENSOR NETWORKS

Qingwen Xu
Old Dominion University, 2005
Director: Dr. Stephan Olariu

A Wireless Sensor Network (WSN, for short) consists of a large number of very small sensor devices deployed in an area of interest for gathering and delivery information. The fundamental goal of a WSN is to produce, over an extended period of time, global information from local data obtained by individual sensors. The WSN technology will have a significant impact on a wide array of applications on the efficiency of many civilian and military applications including combat field surveillance, intrusion detection, disaster management among many others. The basic management problem in the WSN is to balance the utility of the activity in the network against the cost incurred by the network resources to perform this activity. Since the sensors are battery powered and it is impossible to change or recharge batteries after the sensors are deployed, promoting system longevity becomes one of the most important design goals instead of QoS provisioning and bandwidth efficiency. On the other hand the self-organization ability is essential for the WSN due to the fact that the sensors are randomly deployed and they work unattended. We developed a self-organization protocol, which creates a multi-hop communication infrastructure capable of utilizing the limited resources of sensors in an adaptive and efficient way. The resulting general-purpose infrastructure is robust, easy to maintain and adapts well to various application needs. Important by-products of our infrastructure include: 1) Energy efficiency: in order to save energy and to extend the longevity of the WSN sensors, which are in sleep mode most of the time. 2) Adaptivity: the infrastructure is adaptive to network size, network topology, network density and application requirement. 3) Robustness: the degree to which the infrastructure is robust and resilient. Analytical results and simulation confirmed that our self-organization protocol has a number of desirable properties and compared favorably with the leading protocols in the literature.

To my wife

ACKNOWLEDGMENTS

This work could not be completed without the help of many individuals, to whom I would like to express my appreciation. First and foremost, I would like to thank my advisor, Dr. Stephan Olariu, who has put a great deal of time and effort into the guidance of this work.

Next, I would like to convey my sincere thanks to the other members of my dissertation committee, Drs. Larry Wilson, Irwin Levinstein, Ravi Mukkamala and Frederic McKenzie. Their expertise, thorough reviewing and valuable suggestions have also led to a greatly improved dissertation.

I am grateful to my family for their encouragement and support. Finally, special thanks to my wife for her understanding and patience during those days required to complete this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

I.1 OVERVIEW

In the past decade, the areas of mobile computing and wireless networks have seen an explosive growth both in terms of the number of services provided and the types of technologies that have become available. Unlike their wired counterparts, most types of wireless networks are rapidly deployable, scale well, and are cost-effective [1, 90, 91]. Recent advances in nano-technology have made it technologically feasible and economically viable to develop a large variety of Micro Electrical-Mechanical Systems (MEMS)– miniaturized low-power devices, referred to as *sensors*, that integrate sensing, special-purpose computing and wireless communications capabilities [45, 111, 120]. It is expected that sensors will be mass-produced making production costs negligible [1, 90, 112, 120]. Individual sensors lack fabrication-time identities, have a non-renewable power supply and, once deployed, must work unattended. A large number of challenging applications ranging from creating smart environments to embedded agile systems are being contemplated that involve a massive random deployment of sensors, numbering in the tens of thousands or even millions [2, 29, 42, 45, 66, 84, 115].

Fig. 1 shows some sensor prototypes built at UCLA. On the left is a radio sensor capable of sensing temperature and light. On the right is a laser sensor capable of sensing temperature and humidity. It is also expected the sensors will become much smaller in the foreseen future.

It is anticipated that aggregating sensors into sophisticated computation and communication infrastructures, called *wireless sensor networks* (WSN, for short)([14, 21, 40, 105]), will have a significant impact on a wide array of applications on the efficiency of many military and civilian applications, such as combat field surveillance, intrusion detection (i.e. detecting unauthorized access to resources), and disaster management [73, 118, 119]. The fundamental goal of a WSN is to produce, over an extended period of time, global information from local data obtained by individual sensors. WSNs process data gathered by multiple sensors to monitor events in an area of interest. For example in a disaster management scenario, a large number of sensors can be dropped from a helicopter. Networking these sensors can assist

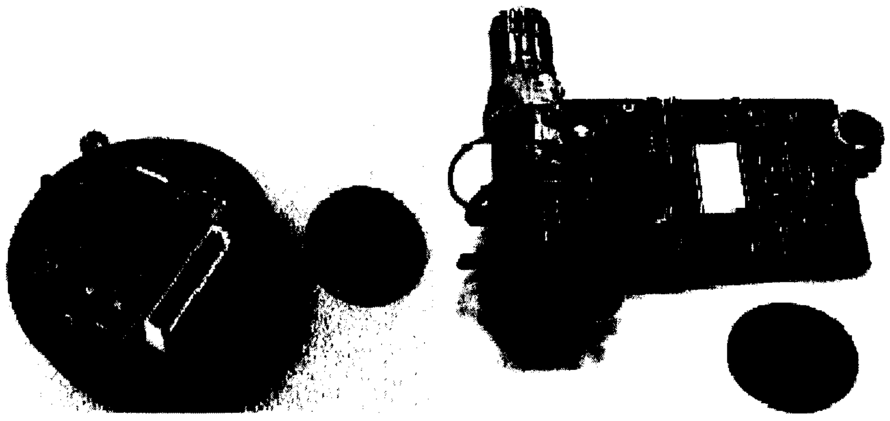


Fig. 1: Sensor prototypes.

rescue operations by locating survivors, identifying risky areas and making the rescue crew aware of the overall situation. On the military side, the use of WSN can limit the need for personnel involvement in the usually dangerous reconnaissance missions [67]. Homeland security applications include law enforcement, remote reconnaissance, monitoring, surveillance and security zones ranging from persons to borders [8, 50, 57, 61, 86, 101].

I.2 THE SENSOR MODEL

Basically, a sensor is an electronic device that is capable of detecting environmental conditions including temperature, sound, or the presence of certain objects. Sensors are generally equipped with data processing and communication capabilities. The sensing circuitry measures parameters from the environment surrounding the sensor and transforms them into electric signals. Processing such signals reveals some properties about objects located and/or events happening in the vicinity of the sensor. The sensor sends such sensed data, usually via radio to a command center either directly or through a data collection station (a base station or sink). The base station can perform fusion of the sensed data in order to filter out erroneous data and anomalies and to draw conclusions from the reported data over a period of time.

For example, in a reconnaissance-oriented WSN, sensor data indicates detection of a target while fusion of multiple sensor reports can be used for tracking and identifying the detected target.

The block diagram of a typical sensor is depicted in Fig. 2. The functionality of the sensing circuitry depends on the sensor capabilities. In general, the sensing circuitry generates analog signals whose properties reflect the surrounding environments. These signals are sampled using the A/D converter and stored in the on-board memory as a sequence of digital values. The sensed data can be further processed using a data processor (microprocessor or DSP) prior to sending them over to the base station using the radio transceiver. The capabilities of the data processor are subject to a trade-off. A powerful DSP can be advantageous since it will allow the sensor to transmit only important findings rather than excessive raw readings. Reducing the sensors traffic generation rate can save the energy consumed by the radio transmitter and can decrease radio signal interference and collisions among the deployed sensors. On the other hand, sophisticated data processing can consume significant energy and can be a cost and a design burden by increasing the complexity of the sensor design. In all cases, the sensor has to include some control logic to coordinate the interactions among the different functional blocks. Such control function can be also performed by the data processor if included.

Sensors are not new at all. For example, the oil industry uses geophone sensors for oil exploration and the military uses arrays of radars for intrusion detection. However, traditional sensor systems are centralized, usually involved a small number of sensors, all wired to a central processing unit where all the information is processed. In contrast, we focus on distributed, wireless sensor network in which the signal processing is distributed along with the sensing. Networking sensors enable the sensors to cooperatively accomplish complex tasks and provide capabilities greater than the sum of individual parts. The sensors are densely deployed either inside the phenomenon or in close proximity hence can gather information that was impractical or expensive to obtain by traditional means. The reason for wireless communication is that in many applications, the environment to be monitored does not have infrastructure for communication, the sensors must rely on wireless communication channel. The reason for distributed processing is that communication is a major energy consumer as the radio power drops off rapidly. Therefore, one wants

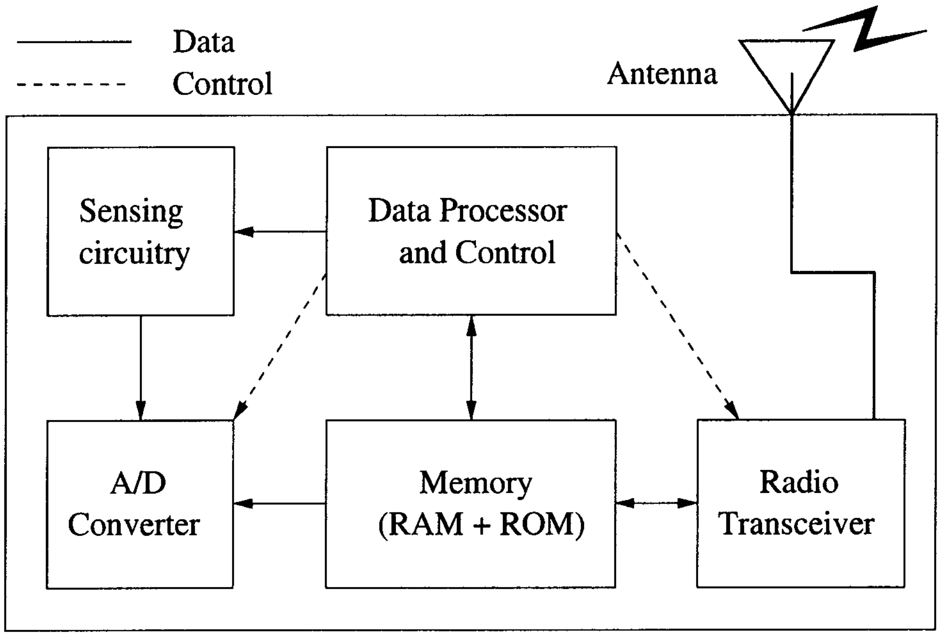


Fig. 2: The block diagram of a typical sensor.

to process the information as much as possible inside the network rather than sending it to a central processing unit. Moreover, instead of sending raw data to some central node, sensors use their computation abilities to locally process the data they gathered (data fusion), and transmit only the required and partially processed data.

We assume a sensor to be a device that possesses three basic capabilities; sensory, computation, and communication. A sensory capability is necessary to acquire data from the environment. A communication capability is necessary for sending (receiving) aggregated data and control information to/from other sensors or the sink. A computational capability is necessary for aggregating data, processing control information, and managing both sensory and communication activity. For our purposes, we abstract each of the above capabilities in terms of operations that the sensor performs. We assume that the unit of activity of a sensor is an operation. At any point in time, a sensor, will be engaged in performing one of a finite set of possible operations, or will be idle (asleep). Example operations are sensing (data acquisition), routing (data communication; sending or receiving), computing (e.g. data aggregation), and roaming (e.g. receiving control data). We assume each operation performed by a sensor consumes a known fixed amount of energy and that a sleeping sensor performs no operation and consumes essentially no energy.

We assume that individual sensors operate subject to the following fundamental operational constraints.

- *Anonymity:* sensors are tiny commodity devices lacking fabrication-time IDs;
- *Non-renewable energy budget:* each sensor has a modest non-renewable energy budget; once the energy budget is exhausted, the sensor becomes in-operational;
- *Sleep-make cycle:* each sensor is in sleep mode most of the time, waking up at random points in time for short intervals under the control of a watchdog timer;
- *Reduced transmission range:* each sensor has a modest transmission range, perhaps a few meters; this implies that outbound messages sent by a sensor can reach only the sensors in its proximity, typically a small fraction of the sensors in the entire network. As a consequence, the WSN must be multi-hop and only a limited number of the sensors count the sink among their one-hop neighbors.
- *Local information:* for reasons of scalability, it is assumed that no sensor knows the topology of the entire network.

I.3 INTERFACING SENSOR NETWORKS

There are several possible techniques that can be used to interface sensor networks to the outside world and, in particular, to harvest the information they produce. Perhaps the simplest involves using one or several sinks, special long-range radios, deployed alongside with the sensors. Each sink has a full range of computational capabilities, can send long-range directional broadcasts to the sensors, can receive messages from nearby sensors, and has a steady power supply. In this scenario, the raw data collected by individual sensors is fused, in stages, and forwarded to the nearest sink that provides the interface to the outside world. Such a scenario, involving three sinks A, B, and C, is illustrated in Fig. 3.

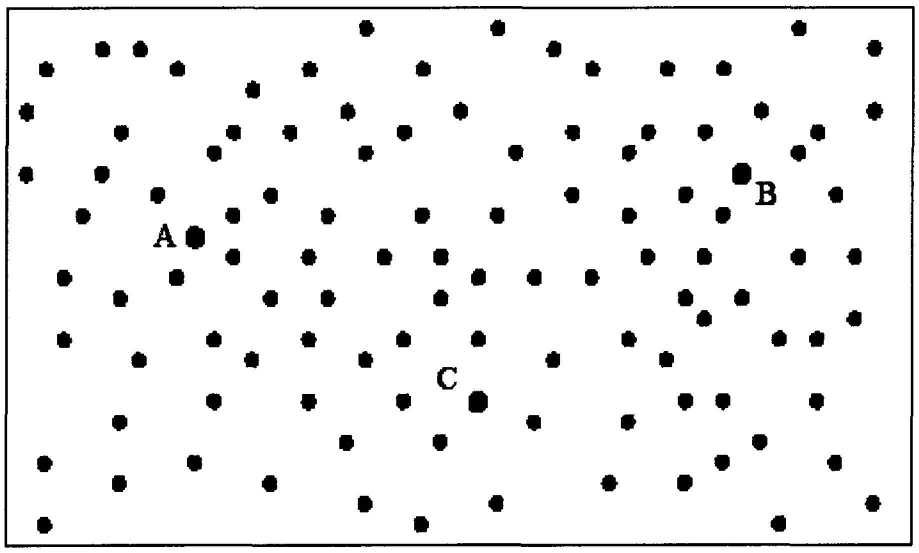


Fig. 3: Illustrating a multi-sink sensor network.

Referring to Fig. 4, we note that the interface with the outside world may be achieved by a helicopter or aircraft over flying the deployment area. In Fig. 4, an external debriefing agent collects information from a select group of reporting nodes (local sinks).

Besides acting as the data traffic destinations, the sinks are also in charge of performing any necessary training and maintenance operations involving the sensor network [75].

I.4 STRUCTURE AND ORGANIZATION OF WSN

Depending on the application, different architectures and design goals/constraints have been considered for WSN. In this section we attempt to capture architectural design issues and to highlight their implications on the network infrastructure and operation models proposed in the literature. We are using the routing protocol for discussion in order to highlight how the infrastructure has been set to fit the network operational model and to deal with the specific architectural issue. As will become

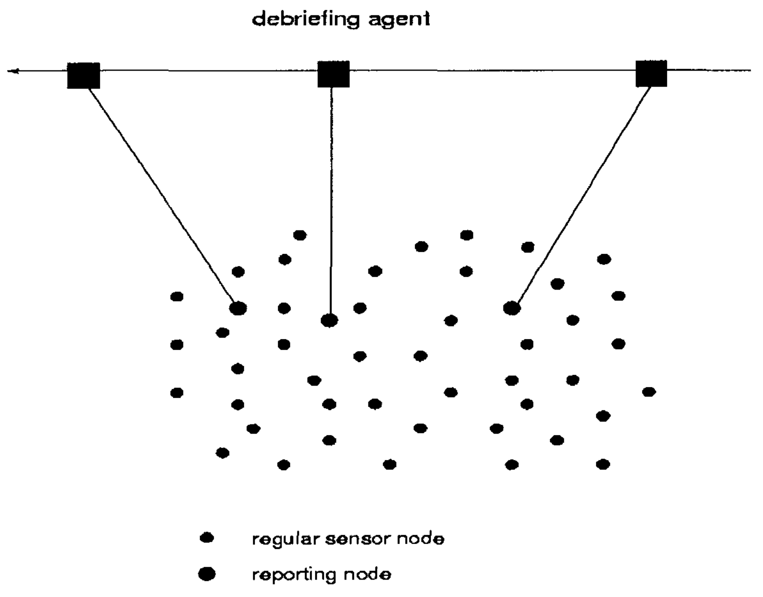


Fig. 4: Information harvesting in a sensor network.

clear, the following short survey demonstrates that the concept of establishing a general-purpose virtual infrastructure that can serve diverse applications and can be mapped to the variant architectural and operational models has not been considered in the literature.

There are three main components in a WSN. These are the sensors, the sink and the monitored events. Aside from the very few setups that utilize mobile sensors, most of the network architectures assume that sensors are stationary. On the other hand, supporting the mobility of sinks or cluster-heads is sometimes deemed necessary. Routing messages from or to moving sensors is more challenging since route stability becomes an important optimization factor, in addition to energy, bandwidth etc. The sensed event can be either dynamic or static depending on the application. For instance, in a target detection/tracking application, the event (phenomenon) is dynamic whereas forest monitoring for early prevention is an example of static events. Monitoring static events allows the network to work in a reactive mode, simply generating traffic when reporting. Dynamic events in most applications require periodic reporting and consequently generate significant traffic to be routed to the sink.

Another design consideration is the topological deployment of sensors. This is application dependent and affects the performance of the communication protocol. The deployment is either deterministic or self-organizing. In deterministic situations, the sensors are manually placed and data is routed through pre-determined paths. In addition, collision among the transmissions of the different sensors can be minimized through the pre-scheduling of medium access. However in self-organizing systems, the sensors are scattered randomly creating an infrastructure in an ad hoc manner. In that infrastructure, the position of the sink or the cluster-head is also crucial in terms of energy efficiency and performance. When the distribution of sensors is not uniform, optimal clustering becomes a pressing issue to enable energy efficient network operation.

During the creation of an infrastructure, the process of setting up the network topology is greatly influenced by energy considerations. Since the transmission power of a wireless radio is proportional to the squared of the distance or even higher in the presence of obstacles, multi-hop routing consumes less energy than direct communication. However, multi-hop routing introduces significant overhead for topology management and medium access control. Direct routing performs well if all the sensors were very close to the sink. Most of the time sensors are scattered randomly over an area of interest and multi-hop routing becomes unavoidable. Arbitrating medium access in this case becomes cumbersome.

Depending on the application of the WSN, the data delivery model to the sink can be continuous, event-driven, query-driven or hybrid. In the continuous delivery model, each sensor sends data periodically. In event-driven and query-driven models, the transmission of data is triggered when an event occurs or when a query is generated by the sink. Some networks apply a hybrid model using a combination of continuous, event-driven and query-driven data delivery. The routing and MAC protocols are highly influenced by the data delivery model, especially with regard to the minimization of energy consumption and route stability. For instance, it has been concluded in [101] that for a habitat monitoring application where data is continuously transmitted to the sink, a hierarchical routing protocol is the most efficient alternative. This is due to the fact that such an application generates significant redundant data that can be aggregated on route to the sink, thus reducing traffic and saving energy.

In a WSN, different functionalities can be associated with various sensors. In early work on WSN, all sensors were assumed to be homogenous, having equal capacity in terms of computation, communication and power. However, depending on the application, a sensor can be dedicated to a particular special function such as relaying, sensing and aggregation since engaging the three functionalities at the same time on a sensor might quickly drain its energy budget. Some of the hierarchical infrastructures proposed in the literature designate a cluster-head different from the normal sensors. While some networks have selected cluster-heads from the deployed sensors in other applications a cluster-head is more powerful than the sensors in terms of energy, bandwidth and memory. In such cases, the burden of transmission to the sink and aggregation is handled by the cluster-head.

I.5 ROADMAP

Our work focuses on the design of ultra-light self-organization and communication protocols for a class of wireless sensor networks consisting of a large number of sensors randomly deployed in an area of interest. A basic management problem in wireless sensor networks is to balance the utility of the activity in the network against the cost incurred by the network resources to perform this activity. The scarce resource in the network that is of primary concern is energy.

The limited power budget of individual sensors mandates the design of energy-efficient data gathering, fusing, and communication protocols. Recent advances in hardware technology make it clear that a major challenge facing the sensor network community is the development of ultra-lightweight communication protocols for self-organization, network maintenance, data collection and fusion, and routing [2]. In regard to network protocol design, we have significant experience on Internet, mobile ad-hoc networks and cellular networks. There is a great deal of protocols developed for those networks. Unfortunately, many existing protocols are not suitable for WSNs, which possess quite different physical and communication characteristics. More importantly, the application requirements are novel and unique, which requires us to reconsider network protocol design principles and methodologies.

This dissertation begins an introduction to the sensor model and the structure and organization of sensor networks. Chapter II discusses applications and current protocol design technologies of sensor networks. Chapter III discusses the sensor network characteristics, where we provide the requirement analyses. We argue that those requirements are so dramatically different from the existing networks and thus make the WSN design somewhat unique to the others. We summarize the requirements that are going to significantly impact the protocol design and emphasis that energy efficiency is the primary concern. And we discuss the methods we are using to develop energy efficient protocols. Chapter IV presents a novel energy-efficient self-organization protocol for sensor networks. The protocol performs the basic self-organization services including establishing communication links and setting up medium access control scheme. By organizing the sensors, the protocol also constructs a general-purpose infrastructure which supports to design various efficient communication protocols. Chapter V discusses another method for constructing a general-purpose infrastructure, called *sensor training.* By training sensors to learn their grain-coarse locations, a training protocol imposes a dynamic coordinate system on top of the sensor network. Chapter VI presents a routing protocol, which using the infrastructure constructed in Chapter IV to fulfill the requirements described in Chapter III. In Chapter IV, V and VI, the protocol performance analysis and the findings in our simulation are also given. This dissertation ends with conclusions and a discussion of future research directions in Chapter VII.

CHAPTER II

STATE OF THE ART

II.1 THE APPLICATIONS OF SENSOR NETWORKS

The sensor network technology started by a DARPA-sponsored SmartDust program [111], where the sensor network is first defined as:

> A sensor network is a deployment of massive numbers of small, inexpensive, self-powered devices that can sense, compute, and communicate with other devices for the purpose of gathering local information to make global decisions about a physical environment.

By this definition, a sensor network consists of a massive number of very small sensors densely deployed in the area of interest. The sensor network is deployed for gathering information from the environment. Later the DARPA definition of the sensor network is expanded a litter further by the National Research Council:

> Sensor networks are massive numbers of small, inexpensive, self-powered devices pervasive throughout electrical and mechanical systems and ubiquitous throughout the environment that monitor (i.e., sense) and control (i.e., effect) most aspects of our physical world.

Thus, the sensor network not only is considered to gather the information from the environment but also to control the environment.

Since building massively-deployed sensor networks is prohibitively expensive under current technology, small-scale sensor networks were developed in the past few years. These small-scale prototypes support a growing array of applications ranging from smart kindergarten [77, 87, 99] to smart learning environments [23], to habitat monitoring [82, 102], to environment monitoring [19, 24, 62], to greenhouse and vineyard experiments [18, 43], and to forest fire detection [18, 19]. These prototypes provide solid evidence of the usefulness of sensor networks and suggest that the future will be populated by pervasive sensor networks that redefine the way we live and work [90].

There are many different types of sensors including seismic, thermal, visual, infrared, acoustic and radar which are able to monitor a wide variety of ambient conditions. In addition, the sensors also can be used for event detection, event identification, location sensing, and local control of actuators. Due to the varieties of these sensors, a large number of sensor network applications is proposed in the literature, these applications includes military surveillance, health monitoring, environmental sampling, machine diagnosis among many others [2].

II.1.1 Military Application

Wireless sensor networks can be an integral part of military command control, communications, computing, intelligence, surveillance, reconnaissance and targeting (C4ISRT) system [2]. Large scale, low cost and small size sensors can be densely deployed in inhospitable physical environment, such as battlefield or toxic locations. These sensors have the ability to cooperate themselves to accomplish a significant range of tasks, such as surveillance, reconnaissance of opposing forces, targeting or damage assessment etc.

II.1.2 Health Application

Smart sensors, which have created by combining sensing materials with integrated circuitry, are being considered for several biomedical applications such as a glucose level monitor, cancer detectors or retina prosthesis [88]. The requirements for medical sensor systems are biocompatibility, fault-tolerance, energy-efficiency, and scalability. For biomedical applications, the locations of sensors are fixed and the placement can be pre-determined. The power must be carefully controlled to avoid damage to the surrounding tissue. The human body is mostly water and thus has attenuation characteristics similar to water. Extremely high frequencies can't be used.

II.1.3 Home Application

The sensors are so simple and small that it can be buried into almost anything [73]: such as vacuum cleaners, microwave ovens, refrigerators, and VCRs. They can interact with each other and with outside network via the Internet. The applications include personal location system, smart environment etc. Most sensors are not mobile or mobile with low speed. The communication must be of reasonably short range to allow proximity to be inferred from connectivity.

II.1.4 Mobile Sensor Network

Mobile sensor technology is also proposed in the literature [92]. These mobile sensors can form a large-scale, ad-hoc wireless network and cooperates to accomplish varied tasks, such as: troops of low-cost sensors are used to explore and acquire maps of unknown environment. The mobile sensors must be able to avoid objects and move to a certain location. The mobile sensors make the network topology dynamic, and truly ad-hoc. On the other hand, since mobile sensors are more "intelligent" than stationary sensors, they can form a network that maximizes certain characteristic. For instance, the mobile sensors can move to locations of low signal strength to improve throughput along a multi-hop transmission path.

II.2 PROTOCOL DESIGN

Early researches [1] identified the technical challenges for designing sensor networks. For examples, the sensor network must be able to self-organize into a functioning network and self-locate and identify information destinations. The information delivered by the sensor network must be in time and must be accurate in terms of false positives/negatives. Those challenges will influence sensor network operating and network protocol choices.

The researchers realized that the sensor network is fundamentally different from previously studied networks [23, 97]. Comparing with the existing computer networks such as the Internet, the mobile ad-hoc networks and various wired or wireless local area networks, the sensor network presents not only different physical characteristics, but also different overall structure of network applications and services. The applications of the sensor network demand a different set of network services. Some of them remain the same of the existing computer networks but need to be redesigned for novel requirements (e.g. energy efficient). Those services include the medium access control (MAC) and data routing. Some of them raise brand new challenges (e.g. self-organization [97], localization [3, 17], coverage [63, 64]) that do not exist before.

By comparing with the existing computer networks, it becomes clear that most existing protocols do not meet the requirements of the sensor network. The massive deployment of sensors in a sensor network, combined with anonymity of individual sensors, limited power budget and – in many applications – a hostile environment, pose daunting challenges to the design of protocols for sensor networks. For one thing, the limited power budget at the individual sensor level mandates the design of ultra-lightweight communication protocols. Likewise, issues concerning how the data collected by individual sensors could be queried and accessed and how concurrent sensing tasks could be executed internally are of particular significance. An important guideline in this direction is to perform as much local data processing at the sensor level as possible, avoiding the transmission of raw data through the sensor network. Indeed, it is known that it costs 3J of energy to transmit 1Kb of data a distance of 100 meters. Using the same amount of energy, a general-purpose processor with the modest specification of 100-million-instructions/watt executes 300 million instructions [84, 98].

As a consequence, the sensor network must be multi-hop and only a limited number of the sensors count the sink among their one-hop neighbors. For reasons of scalability, it is assumed that no sensor knows the topology of the network.

One of the approaches for designing sensor network protocols is to model the sensor network after conventional computing networks. Based to the great amount of design experiences learned from the Internet and the mobile ad-hoc networks, protocols about self-organization [97], medium access control [116] and routing [7, 41] are designed to meet the requirements of the sensor networks. This approach has led to many successes, however, only on the network communication level. On the application level, the experiences stop helping the protocol design since the nature of the sensor network is about gathering information instead of providing point-to-point communication. This fact forces the researchers reaching out for other methodologies. One of these promising approaches, proposed by Jones et al. [44], is to model the sensor network after the biological ecosystems.

The authors of [44] argued that in the presence of a massive deployment, sensor networks must behave as a community of organisms, where individual sensors operate asynchronously and autonomously in parallel. For that, sensor networks can benefit from lessons learned from the way biological ecosystems are organized. The paper demonstrated that fully distributed data aggregation can be performed in a scalable fashion in massively deployed sensor network. Based on this model, novel techniques for data aggregation, energy conservation and bottleneck elimination are developed.

CHAPTER III

DESIGN REQUIREMENT

III.1 NETWORK CHARACTERISTICS

III.1.1 Physical Characteristics

It is widely believed that the following physical features will dramatically impact the design of sensor network protocols:

- Extremely limited in power, bandwidth and memory
- Random and massive deployment
- Facing highly dynamic situation (in terms of tasks, environment settings, network topology)
- Sensors work unattended
- Prone to failure

First of all, the sensors are battery powered, and it is often impossible to change or recharge batteries after the sensors are deployed. Energy consumption becomes of primary concern for designing protocols. In order to show a clear picture, let us consider the sensor operations described in Section I.2. The major tasks performed by a sensor are: sensing, computing and communication. Research shows that in a low power radio network, the communication (routing, roaming and idle) consumes much more power than computing and sensing. Moreover, the energy consumed by receiving and listening (attempting to receive) is of the same order of magnitude as transmitting. Typically, in an existing ad-hoc wireless network, the idle-receive-send ratio of energy consumptions is about 1:220:300. In a lower energy radio network such as the DEC Roamabout radio network [47], the above ratio is reduced to 1:8.23:16.2. Stemm and Katz [100] reported that the ratio is further reduced to 1:1.05:1.4 in a very low power radio network. Therefore, the major challenge becomes to design network protocols that minimize the communication operations, which, in turn, indicates that sensors should be in sleep mode most of the time. Since the sensors are massively deployed in an area, normally it is not necessary that the sensors monitor the environment all at the same time. At any moment in time, only a small set of sensors (referred as *the active workforce*) is required to be awake to monitor the environment. On the one hand, we want the sensing field provided by the active workforce covers an area as large as possible. On the other hand, the active workforce is desired to be as small as possible for prolonging network lifetime.

Second, Sensor networks face highly dynamic situations in the sense that sensors are normally deployed on a remote terrain or in a hostile environment. Sensors can die for many reasons: Energy depletion, being destroyed or mechanism failure. During the network lifetime, sensor failure could be a regular event. Unlike the network nodes in the Internet, sensors are not given individual attention due to their tight environment constrains. The sensors must work unattended. Moreover, the sensors work together as an integrated system to gather and process information. The sensors must coordinate to establish a communication network after the deployment and adapt to any configuration changes. And all those tasks have to be done without human intervention.

Another important physical feature is that the sensor network normally uses the wireless communication. The wireless communication is much more difficult to achieve than the wired communication because the surrounding environment interacts with the signal, blocks signal paths and introduces noise and echoes. As a result, the protocol design faces more difficulties such as low bandwidth, high error rates, and frequent disconnections. Furthermore, the wireless communication is a broadcast communication in nature. The challenges for protocol design also include the well known near-far problem, hidden terminal problem and the broadcast storm issue [71]. Note that wired sensor networks are also proposed in the literature. Wired sensor networks can be used in health caring, smart home, and machine diagnostic etc. However, many sensor network applications only work with wireless sensors because of the environmental constrains. For instance, sensors may be deployed in the battlefield, forest, mountain or space, where the wired sensor network is impossible to establish.

III.1.2 Communication Characteristics

Application-specific

Most existing computer networks such as the Internet and MANET are application independent communication networks. Although those networks are somewhat different based on their physical features and communication manners, from application point of view, they are very similar systems. That is, the network is an end-to-end communication system, which is designed to be independent of various applications.

To achieve the application independency, a set of design principles called "end-to-end arguments" were established. The reasons behind the principles were stated very clear in [6]:

> The end-to-end arguments concern with building a general-purpose communication system. The principles suggest that specific application-level functions usually cannot, and preferably should not, be built into the lower levels of the system.

These principles indicate that applications live at the "endpoints" of the networks, where the application data are processed. The network communication services are responsible for forwarding packets between the "endpoints" of the network. These principles are realized very well by the layered architecture model (OSI and TCP Reference Model). In this model, not only are the applications independent to the network communication services, layers are also independent to each other. Each layer specifies its own interface (a set of input and output services) to its users (other layers). In this way, the whole system is partitioned into several software components (layers). Those components are able to work together regardless how they are implemented as long as their interfaces are the same. Each component is able to evolve individually. The layered architecture design model dramatically decreased the complexity of network software development and maintenance. However, an important drawback of the general-purpose communication system is that it is impossible to optimize network resource utilization for all the applications. And more than likely, independent application usually implies that the network resource utilization is not optimized for any of the applications.

Due to the extreme limitations in power, memory, communication and computation capabilities, the advantages of the application-independent are thwarted by the energy efficiency requirement. Fundamentally, the sensor network is a new class of network, which we called an *application-specific data gathering, processing and delivering network* to distinguish with the traditional *end-to-end packet switching network* ([13]). As it is said in [23]

> Traditional networks are designed to accommodate a wide variety of applications. Sensor networks are tailored to the sensing task at hand.

The application-specific nature of the sensor network indicates that, instead of simply forwarding packets, the sensor network is specified how to process and deliver data. Such requirements include the sensing task, deadline, location, priority, and cost etc. Application level information is involved into the low-level communication protocols. The application information involvement demands to design so called *data-centric* protocols. On the other hand, traditional networks are packet switching networks, where the application information is embedded in the packets and will not be processed by the low level communication protocols.

For instance, data gathered by sensors have to be processed locally and sent to a "local" base station for data fusion. If the raw data were sent to the "end" user, the communication overhead would be excessive, far exceeding the restrictions imposed by low power and low bandwidth communication. And in many cases, sensors have to process the raw data cooperatively to produce results such as the location, speed and direction of a moving target.

Communication mode

One of the critical differences between WSNs and conventional computer networks is that the former does not operate in point-to-point mode. Generally speaking, there are only two types of traffic in the sensor network.

- Information querying (one to many);
- Information delivering (many to one)

Queries are sent by the sink to task the sensors, request for information, or order to collect more data. Flooding or limited flooding mechanisms are usually employed to propagate queries in the network (one to many mode). And the requested data are sent by the sensors all the way back to the sink to answer the query (many to one mode). This feature has a dramatic impact on the data routing protocol design. In traditional computer networks, which operate in point-to-point mode, routing protocols need to maintain routes for any pair of nodes in the network. Indeed, popular routing protocols existing today in the Internet are mainly based on all sources (destinations) shortest path algorithm. It is clear that those protocols will not migrate well to the sensor network not only because they are heavyweight in terms of network resources but because they are inappropriate. Also, because of the sheer number of sensors, the sensor network requires an efficient flooding mechanism for information querying. A so called "broadcast storm" problem described in [71] demonstrates the flooding deficiencies. Tackling this problem is more about establishing a communication infrastructure and less about designing algorithms.

Another important difference between the sensor network and the traditional computer network is that sensors do not need addresses (e.g. MAC address and IP address). In the traditional computer network (e.g. Internet), node address is used to identify every single node in the network. Various communication protocols and algorithms are based on this low level naming scheme. However, the sensor network is about information retrieval, not point-to-point communication. That is, the sensor network applications focus on collecting data, rather than providing communication services between network nodes. Individual sensor addresses is not essential for WSN applications [36].

To emphasis the differences, we summary the unique characteristics of the sensor network here:

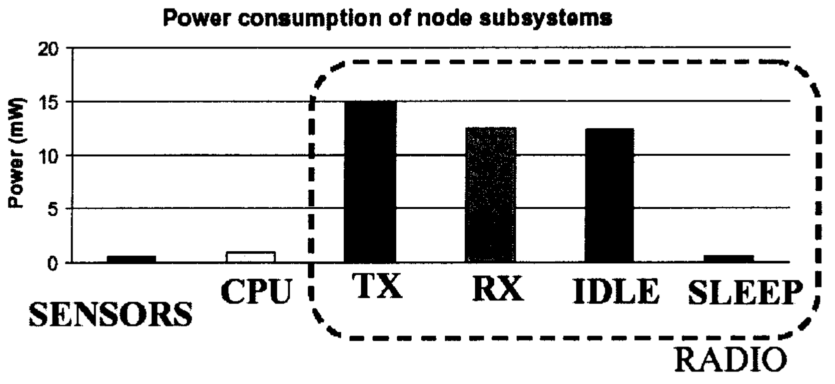
- The sensor network is an application specific data gathering and delivering network;
- Sensors use broadcast communications instead of point-to-point communications;
- Sensors do not need and thus normally do not have global identifications.

III.2 DESIGN REQUIREMENT

III.2.1 Promoting System Longevity

Due to the power limitation and lack of means for changing or recharging batteries, promoting system longevity is the most important requirement for the WSN design.

Power consumption becomes the primary consideration instead of QoS provisioning or bandwidth efficiency [98]. In order to save energy, we need to identify the energy consumers first. According to the sensor model described in Chapter I, we can think of a sensor that consists of three components: the sensing unit, the communication unit and the processing unit. Although the power consumption can vary with different application tasks, it is concluded that the communication unit is the biggest energy consumer on board. Fig. 5 shows the typical power consumption of sensor subsystems. One can see that the communication unit dominates the power consumption. Moreover, main power consumption of communication is for transmission, receiving and idle listening (trying to receive). Surprisingly, the idle listening consumes the same amount of energy as receiving. Hence, leaving the transceiver on (i.e. idle listening) for long periods will be the major factor that impacts the longevity of the network [98].



- $E_{TX} \approx E_{RX} \approx E_{IDLE} >> E_{SLEEP}$
- **Need to shutdown the radio**

Fig. 5: Identify power consumption.

III.2.2 Scalability

One of the major concerns in designing protocols for the WSN is scalability. We expect there are a large number of sensors in the WSN and these sensors are densely deployed. A desired protocol should not only scale to the number of sensors but also to the network density [29]. The network density $\mu(R)$ is defined in terms of number of sensors per nominal coverage area. Thus, if N sensors are scattered in a region of area A, and the transmission range of each sensor is R,

$$\mu(R) = \frac{N \cdot \pi \cdot R^2}{A} \tag{1}$$

Network density is a very important parameter in sensor networks. In [29], the authors explore some principles for designing a scalable, long-lived, robust and self-organizing sensor network. They point out that sensor network algorithms are more effective when the network is dense. However, a denser network poses challenges for network protocol design. For examples, radio transmission is more likely to be collided and interfered. Also, it will cause more overhead for building and maintaining routes in the network.

III.2.3 Self-organization and Self-maintenance

The self-organization capability is essential for the sensor network due to the fact that the sensors are random deployed and they must work unattended. The goal of a self-organization protocol is to create a communication infrastructure that:

- supports various communication protocols, and
- allows the protocols to utilize the network resources efficiently.

The infrastructure built by the self-organization protocol can have significant impacts on WSN design: A hierarchical structure is preferred to data fusion and scalability requirement; a backbone infrastructure helps to alleviate the "broadcast storm" problem and reduces the control message overhead for data routing; and a cluster structure can ease the tasks for medium access control and active workforce selection.

However, the creation and maintenance of the infrastructure is also a burden to carry. On the one hand, it is an overhead in terms of power, bandwidth and latency that should be minimized. On the other hand, due to the sensor failure, the infrastructure must be able to self-maintain efficiently for a long time period.

III.2.4 Adaptivity

Sensor networks face a highly dynamic environment. Such dynamics include power level, task, network density and network topology etc. The protocols designed for sensor networks should allow the communication system to adapt to those dynamics. Furthermore, a sensor network is likely a redundant system. Since a large number of sensors are densely deployed, neighbor sensors could be very close to each other. Also the sensing region of these neighbor sensors could be highly overlapped. Hence the sensors can produce highly redundant information. The effects of the redundancy are twofold: On the one hand, redundant information occupies network bandwidth, decreases system throughput and consumes energy unnecessarily. On the other hand, redundant information increases the fault-tolerance of the system and improves the fidelity of the answers. The challenge (or trade-off) is to maximize the advantages of redundancy and minimize its disadvantages.

III.2.5 Secondary Requirements

Other requirements such as QoS, bandwidth efficiency, throughput, latency and fairness, which are the primary concern in the other networks, will become secondary in the sensor network. Nevertheless, these secondary goals still need to be taken into consideration for protocol design. For instance, in MAC protocol design, the fairness is an important goal in conventional networks since behind each network node there is a human user. It is desired to give each user equal opportunity and time to access the medium. In the sensor network, the fairness is not a major concern as long as the whole system works well. The major goal of a MAC protocol is to provide energy efficient medium access. However, a fairness system is usually a better one in respect with prolonging network lifetime. Intuitively, a sensor network in which each sensor equally consumes energy will be alive longer than a sensor network that does not.

III.3 METHODOLOGIES

III.3.1 Localized Algorithms

In a thought-provoking paper, Estrin et al. [23] argued that the basic characteristics of WSNs, including fully distributed operation and a highly dynamic topology, make it imperious to design protocols that are localized rather than centralized. Centralized protocols require global information at each sensor. Consequently, these protocols may perform well only for small networks. In fact, in large WSNs global information can either be hard or even impossible to obtain in a timely and energy-efficient fashion.

On the other hand, localized protocols are especially attractive for WSNs because of their scalability and robustness. They only require local information which, as a rule, is readily available to individual sensors by virtue of data collection, data fusion, and strictly local communication with immediate neighbors. A number of protocols in the literature are localized, but use an excessive number of messages between neighbor sensors. For instance, some topology control and position determination protocols require a large number of messages to be exchanged between neighbors. Because of the severely limited bandwidth, energy budget and medium access problems caused by excessive messaging, message exchanges between neighbors to construct and/or maintain a local topology or to perform any other operation should be minimized, possibly avoided altogether. In addition to localized protocol operation, it is also important to consider the maintenance cost of such topology. For instance, if the cluster structure is adopted, what happens when cluster leaders move or fail? Does the update procedure remain local, and, if so, what is the quality of the maintained structure over time? Some maintenance procedures may not remain local. This happens when local change triggers message propagation throughout the network. Of course, localized maintenance is preferred, meaning that local topology changes should be performed by a procedure that always remains local, involving only the neighborhood of the affected sensors. Occasionally, local information may be supplemented by a limited amount of global information broadcast by the sink (for instance, the position of the sink). To suit this scenario, Chan and Perrig [12] call a protocol strictly localized if all information processed by a sensor is either local in nature or global but, in this latter case, obtainable immediately (in short constant time) by querying only the sensors immediate neighborhood. As an illustration, consider a protocol that constructs a spanning tree by performing a distributed Breadth-First Search involving only local communications. Such a protocol would be a localized protocol but not a strictly localized since it takes time proportional to the diameter of the network and the entire network must be traversed before the spanning tree can be constructed. This definition of strictly localized protocols captures the ability of localized protocols to perform independent and simultaneous operations which is especially desirable for WSN.

III.3.2 Virtual Infrastructure

Overlaying a virtual infrastructure over a physical network is a time-honored strategy for conquering scale. There are, essentially, two approaches to this exercise. The first is to design the virtual infrastructure in support of a specific protocol. However, more often than not, the resulting infrastructure is not useful for other purposes. The alternate approach is to design a general-purpose virtual infrastructure with no particular protocol in mind. The challenge, of course, is to design the virtual infrastructure in such a way that it can be leveraged by a multitude of different protocols.

Along this line of thought, our strategy is to design a simple self-organization protocol for massively deployed WSN consisting of a large number of anonymous, energy-constrained sensors. An interesting by-product of our self-organization protocol is a robust virtual infrastructure. Importantly, in addition to being strictly localized our self-organization protocol is energy-efficient. Extensive simulation results show that our self-organization protocol has a number of desirable properties and compares favorably with the leading protocols in the literature. In particular, the resulting general-purpose infrastructure is robust, easy to maintain, and adapts well to various application needs. One of the virtues of this infrastructure is the creation of a powerful multi-hop communication network capable of utilizing the limited resources of sensors in an adaptive and efficient way. Additional important by-products of our virtual infrastructure include:

- Energy efficiency: In order to save energy and to extend the longevity of the WSN, sensors keep in sleep mode most of the time;
- Adaptivity: The resulting virtual infrastructure is adaptive to network size, network topology, network density, and application requirement;
- Robustness: The degree to which the virtual infrastructure is robust and resilient.

III.4 SUMMARY

Early computer networks were designed with the hardware limitations as the main concern and the software as an afterthought [104]. After the hardware become mature and stable, this strategy no longer works. Highly complicated software was developed for the computer networks. The WSN technology is very much in its infancy stage. Again, we have to go back to the old time: designing the protocols with the hardware limitations as the main concern. They include the power, communication, computation and memory limitations. Besides those limitations, we face a totally new issue: the WSN is an application-specific data gathering network, not a general-purpose communication network. This issue raises completely different system design philosophies. We are on an uncharted territory now. The biggest questions we are facing are-what's the software architecture for the sensor network, what are the new system design principles? In this dissertation, we developed several protocols and algorithms for sensor network self-organization, location training and data routing. Hopefully, we can gain some experiences for designing sensor network and find some clues for answering above questions.

CHAPTER IV

SELF-ORGANIZATION

In this chapter, we consider the sensors to be randomly scattered in an area of interest. After the deployment, a self-organization protocol is required to build a communication infrastructure. The infrastructure provides a basic platform for

- *Hop-by-hop wireless communication*: The communication links between the sensors have to be established.

  The existing ad hoc networks usually use the neighbor discovery mechanism to establish communication links. As it turns out, the neighbor discovery mechanism is not necessary for the sensor network and its communication overhead can be avoided.

- *Medium access control*: The scheme for sharing the communication medium has to be specified.

  The medium access control scheme is the determining factor of the sensor network lifetime and thus needs to be carefully designed.

- *Sensor scheduling*: The sensors have to divide the sensing task among themselves.

  The most efficient manner of utilizing sensor network resources is to schedule the sensors for the sensing and communication tasks.

IV.1 BACKGROUND

IV.1.1 Medium Access Control

In this section, we investigate various existing MAC protocols designed for ad-hoc wireless networks.

In all the shared-medium networks, the medium access control (MAC), which provides a fundamental network service, decides when and how the neighbor nodes transmit and receive packets to/from each others. The main objective of the MAC protocol is to avoid collisions so that two interfering transmissions do not occur at the same time. The collision causes energy and bandwidth to be wasted due to corrupted packets and subsequent retransmissions. Even worse, collision detection is not possible in a radio communication network due to what is known as the near/far problem ([60]):

> To detect a collision, a station must be able to transmit and listen at the same time, but in radio systems the transmission drowns out the ability of the station to hear a collision.

Another MAC-layer problem specific to wireless is the hidden terminal issue ([46], which also can cause collisions. The hidden terminal problem is illustrated in Fig. 6: When node A begins transmitting a packet to node B, since Node C is out of the range of node A, it thinks the media is free. So node C may start to transmit a packet at the same time or little later. The two packets are collided at node B.

The hidden terminal problem shows that, without an organized structure, collisions avoidance is not a purely local problem. Decisions made by nodes that are two-hops away can affect each other. And it is also possible that the decisions cause a chain reaction that produces global effects.

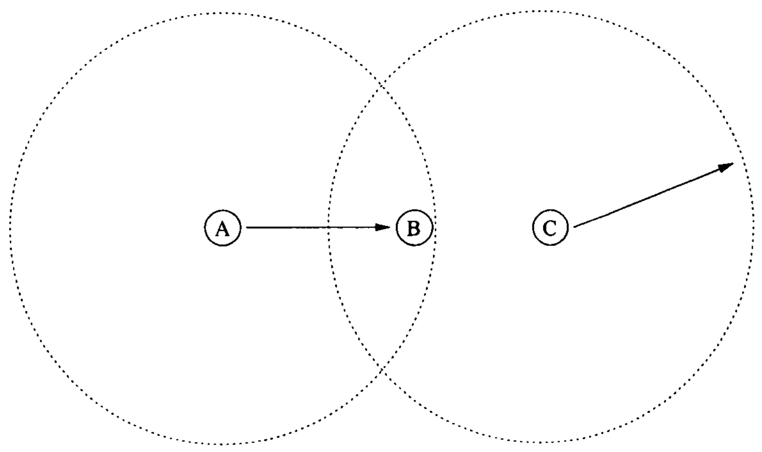


Fig. 6: Hidden terminal: node A is hidden from node C.

There are many MAC protocols that have been developed to avoid or prevent collisions. Table 1 shows two major research lines that dominate wireless network MAC protocol design.

Table 1: MAC protocol category

Category	Representative network
Contention based protocol	Mobile ad hoc networks
Schedule based protocol	Cellular networks

A typical example of the contention based medium access control protocol is the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol. Basically, in every CSMA based protocol, network nodes attempt to avoid collisions by sensing the media. The CSMA/CA scheme uses a two-way handshake procedure (DATA/ACK). Each node senses the media before transmitting the DATA packet. If the media is free, the DATA packet is transmitted. Due to the near/far problem, the sender is not able to detect DATA packet collision. An explicit acknowledgement (ACK) packet is required to be sent by the receiving node to conform that the DATA packet arrived intact. If the media is busy, the node with DATA packets will delay the transmission until the media is free (based on some back off mechanism).

IEEE802.11 MAC is a standard CSMA based MAC protocol, which also specifies an optional control mechanism (RTS/CTS) to solve the hidden terminal issue. According to this scheme, the control packets (RTS/CTS) temporally reserve the media and prevent the interfering transmissions.

PAMAS [94] modifies IEEE 802.11 protocol to save energy. The basic idea is that a node powers off if it is overhearing a transmission and does not have a packet to transmit. However, it requires an extra radio system and more importantly, it does not address the issue of reduce idle listening, which is the dominating factor for energy conservation to a low power radio communication system.

Basically, since CSMA based MAC protocols require network nodes monitoring (idle listening) the channel all the time, from energy conservation point of view, they are not the best choice for the WSNs.

Recent progress to achieve energy efficiency in contention-based MAC protocols result in protocols that hybrid the CSMA with a time frame, such as SMAC [116] (discussed in the next section) and IEEE 802.15.4 [32].

IEEE 802.15.4 is a developing standard for low-complexity, very low-power and lower cost wireless personal area networks. IEEE 802.15.4 MAC requires that each node send a beacon packet once a frame. After sending the beacon, a node will listen to a channel for a very short time period to allow other nodes make contact with it and otherwise turn off the radio. CSMA/CA scheme is used to avoid collisions when multiple nodes contend for the same destination.

Although energy efficiency can be achieved in a contention-based MAC protocol, there are other issues that prevent them from being employed in sensor networks, as it is pointed out correctly in [2]:

> Traditional CSMA based schemes are deemed inappropriate as they all make the fundamental assumption of stochastically distributed traffic and tend to support independent point-to-point flows. On the contrary, the MAC protocol for sensor networks must be able to support variable, but highly correlated and dominantly periodic traffic.

Another major research line of MAC protocols are based on reservation and scheduling. Typically, the TDMA frame is established at local in a group of nodes called a cluster. To reduce the likelihood of power interference, each cluster operates on different code or frequency. This class of MAC protocol is inspired by the cellular network communication model. The self-organized version was first proposed in [4] for a small radio network. The basic idea is refined by M. Gerla and colleagues [30] and converted into a widely recognized medium access control framework. Various protocols based on this framework were developed, such as the MAC protocol in commercial wireless communication product (Blue Tooth [33]), and the LEACH [38] in sensor networks as well.

Blue Tooth is the combination of TDMA in piconets (clusters) and frequency-hopping (FH)-CDMA. The piconets are dynamic established and released. Only eight nodes are allowed in a piconet. A special mechanism is developed to establish connections between nodes. In order to find each other and make connection, three elements have been defined to support connection establishment: scan, page, and inquiry. A node in idle mode wants to sleep most of the time to save power, in order to allow connections to be formed, the node periodically wakes up to listen for its identity. A node that wants to connect has to transmit the access code (derived from callee's identity) repeatedly at different frequencies. To save energy, a paging unit is added on each node.

Schedule based protocols have a natural advantage of energy conservation compared to contention based protocols, because the duty cycle of the radio can be reduced significantly and there is no contention-introduced overhead and collisions. However, managing a hierarchical structure in a dynamic network and supporting inter-cluster communication are not easy tasks. Both LEACH and Blue Tooth use a simple two-level hierarchical structure. In LEACH, the clusters do not communicate with each other, instead, the cluster leaders communicate with a sink (the destination of the sensing information) directly.

IV.1.2 Related Work on Self-organization

The main goal of this section is to offer a succinct review of some of the self-organization protocols proposed in the recent literature on WSN. These protocols are typically used to organize the network into several clusters [2, 5, 12, 16, 28, 107, 110] or to construct a spanning tree [39, 55, 58, 121] infrastructure for data collection and delivery. There are other approaches as well. In [42], the virtual infrastructure consists of a set of paths dynamically established as a result of the controlled diffusion of a query from a source node into the network. Relevant data is routed back to the source node, and possibly aggregated, along these paths. The paths can be viewed as a form of data dissemination and aggregation infrastructure. However, this infrastructure serves purpose of routing and data aggregation and it is not clear how it can be leveraged for other purposes. A similar example is offered by [9] where sensors use a discovery procedure to dynamically establish secure communications links to their neighbors; collectively these links viewed as a secure communications infrastructure. As before, it is not clear that the resulting infrastructure be leveraged for other purposes. Quite recently, Olariu et al. [75] have proposed a powerful virtual infrastructure that is general purpose and that can be leveraged by a large number of protocols to provide energy-efficient solutions to an array of applications ranging from data warehousing to security. However, the virtual infrastructure on [75] assumes WSN is already self-organized.

Clustering was proposed in large-scale mobile networks as a means of achieving scalability through a hierarchical approach. For example, medium access layer, clustering helps increase capacity by promoting the spatial reuse of the channel; at the network layer, clustering helps flooding efficiently, reducing the size of routing tables, and striking a balance between reactive and proactive routing control overhead. Unfortunately, the vast majority of clustering protocols designed for MANET and other wireless network do not migrate well to WSN. Indeed, most of the clustering protocols for wireless networks (e.g. lowest ID, highest connectivity [30], weighted clustering) rely on neighborhood information. To collect neighborhood information, the clustering algorithms have a neighbor discovery phase before the clusters are constructed. A Hello message is used to exchange information between neighbors. To ensure the correct collection of neighborhood information, the "Hello" message has to be broadcasted repeatedly. In WSNs, the "Hello" message mechanism will not work well for several reasons. If the sensors are in sleep mode most of the time, it takes much more time and energy for sensors to collect neighborhood information. The denser the network, the larger the probability that "Hello" messages will collide, which costs even more time and energy. Accurate neighbor information is critical in point-to- point communication networks. It is needed not only for clustering, but also for some more fundamental network functionality such as routing. This is not necessarily true in WSNs.

Due to the importance of clustering a number of lightweight clustering protocols have been proposed for WSNs [5, 38]. In these clustering protocols, cluster leaders are elected at random: sensors elect themselves to be leaders with a predetermined probability p. The resulting protocols tend to be energy-efficient: they do not rely on neighbor discovery; and, in addition, they scale in both the number of sensors in the network and network density.

However, by relying on probability p as a fixed system parameter is way to rigid making the protocol un-adaptive to WSN dynamics. When the network topology changes (e.g. sensors expire or fresh sensors are added), the probability has to be changed accordingly. Usually it is not easy to do so in a large, multi-hop and dynamic network. In addition, random leader election is somewhat arbitrary. The elected leaders could be too crowded in one area and too sparse in another area. Moreover, communication between clusters is not fully addressed in those algorithms. In a multi-hop, cluster-based communication system, an efficient and adaptive mechanism for inter-cluster communication is necessary.

NeuRFon netform [39] is a self-organizing wireless network for low data rate, low-power sensors. A low duty cycle MAC scheme like IEEE 802.15.4 is used to reduce energy consumption at each sensor. At network formation time, a spanning tree backbone is constructed to support multi-hop routing. In [39], network maintenance algorithms are also proposed to help maintain network cohesiveness. One issue of spanning tree self-organization is a long network formation period. The simulation results in the paper [39] showed that a 64-nodes network needed approximately 95s for network formation. The reason of a long formation period is because the spanning tree formation algorithm is not a strictly local algorithm. It indicates that the algorithm is not scalable to network size and density.

S-MAC [116] puts sensors into periodic sleep mode for energy conservation. Each sensor is free to choose its own listen/sleep schedules (periodically sleep and listen) and broadcast the schedule to all its one-hop neighbors. Some neighbor sensors may choose the same schedule to form a virtual cluster. A sensor can receive packets during its listening period. A contention MAC mechanism (same as IEEE 802.11) is used when multiple sensors want to talk to a sensor.

Sohrabi and Pottie [96, 97], Sohrabi et al. [98] and Pottie and Wagner [84] have proposed various self-organization protocols for WSNs. The most intriguing of these, reported in [98], that we shall refer to as SMACS, combines neighbor discovery and channel assignment phases. A sensor wakes up at random times and sends invitation messages to find its neighbors. After a sensor finds a neighbor, these two sensros negotiate and assign a pair of slots on the super frame (TDMA) for transmission and reception. The basic idea is let sensors form links on the fly. They call it non-synchronous scheduled communication (NSC).

LEACH [38] is a clustering-based protocol that utilizes randomized rotation of local cluster leaders to evenly distribute the energy load among the sensors in the network. The algorithm is run periodically to ensure every sensor becomes a leader at least once within 1/P rounds, where P is the desired percentage to become a leader in each round. Data fusion can be used to reduce global communications. The authors point out that using a minimum energy routing protocol, sensors that are near the sink will die fast. In LEACH, cluster-leaders send data directly to the sink to prolong the network lifetime. However, it limits the protocol's usage because in a very large network some sensors may be deployed far from the sink and not able to directly communicate with the sink.

In [107], the necessary organization techniques for WSNs are identified. These techniques include: sensor deployment, sensor activation, neighbor discovery, cluster formation, routing, and network maintenance. Among these techniques for network organization, neighbor discovery and cluster formation were investigated in the paper. The author pointed out that efficient neighbor discovery can be a challenge due to the fact that sensors are in sleep mode most of the time. A modified beacon approach (advertise-reply beacons) was proposed to discover neighbors efficiently. Nevertheless, neighbor discovery is an energy consuming procedure, especially when the network is dense.

The idea of clustering in [107] is to periodically have sensors volunteer to elect themselves as cluster leaders if volunteers are qualified. The leaders ask their members to campaign for new members until an adequate group size is reached. The author claims that this approach is adaptive to the network topologies and is realizable in a relatively simple state machine that does not incur heavy communication patterns.

IV.2 SELF-ORGANIZATION PROTOCOL

The main goal of this section is to spell out the details of a novel self-organization protocol that will endow the amorphous set of massively deployed sensors with a robust virtual infrastructure.

Ideally, we desire that sensors only power on when they have works (sensing, receiving or transmitting) to do. To achieve energy efficiency, sensors' awake time should be scheduled. For a large scale sensor network, global scheduling is out of question. The natural way is to organize sensors into clusters. In each cluster, a cluster leader can schedule the sensors' sleep and awake time to avoid idle listening and to achieve collision-free communication. The cluster leader can specify at what time and which sensors in the cluster are going to sense, receive or transmit. The other sensors of the cluster are free to power off if there is no work assigned to them.

When the sensors are organized into clusters, application tasks are assigned to clusters, not individual sensors. The cluster leader can have the tasks accomplished adaptively depending on the cluster conditions (e.g. the size of the cluster, power levels etc.) and the task requirements (e.g. reliability, latency etc.). Inside a cluster, the leader can arrange sensors to consume energy equally and to communicate collision free. And if neighbor clusters use different frequency for transmitting, power interference between clusters is also reduced.

Our self-organization protocol organizes a large number of sensors into a multi-hop, collision free and adaptive communication infrastructure. After the infrastructure is constructed, energy efficiency can be achieved via local scheduling. Three main functionalities of the protocol are listed below:

1. Establishing a multi-hop communication infrastructure

   The constructed infrastructure enables the collision free communication. On the one hand, the infrastructure isolates clusters to reduce the power interference (neighbor clusters have different frequency channels). On the other hand, the collision resolution mechanism avoids the power interference between neighbor leaders (neighbor leaders have different beacon sending time).

2. Selecting the sensing workforce in each cluster

   Based on the sensing task at hand, a set of sensors are selected as an active workforce in each cluster. In a cluster, the leader schedules the routine transmitting and receiving events. Unpredictable events are handled by the wakeup mechanism.

3. Maintaining the communication infrastructure

   In order to provide robust and continuous services, a leader retirement scheme is developed for maintaining the clusters. In addition, unpredicted leader failures are handled by re-clustering mechanism.

IV.2.1 Clustering

Our leader election scheme is based on the club algorithm developed in [69]. After deployment, each sensor sleeps for a random time period uniformly distributed over an interval $(0, T)$, after which it waking up. Upon waking up, each sensor starts to listen for a *beacon frame* time period. We note here that the length F of a beacon frame is a system parameter determined a priori as a function of the overall mission of the deployment. If a sensor has not received any beacon packets by the end of its listening time period, it becomes a leader and starts to broadcast beacon packets periodically as showed in Fig. 7(a). The leader broadcasts a beacon packet every beacon frame to announce its leadership. If a sensor received some beacon packet by the end of its listening time period, it becomes a cluster member of some cluster. A member uses the beacon sending time to distinguish neighbor leaders and it has to be awake at its leaders' beacon sending time to receive the beacon packet. To ensure that the sensors are receiving all the beacon packets, a member repeats its random sleep pattern (Fig. 7(b)) until leader election phase ends.

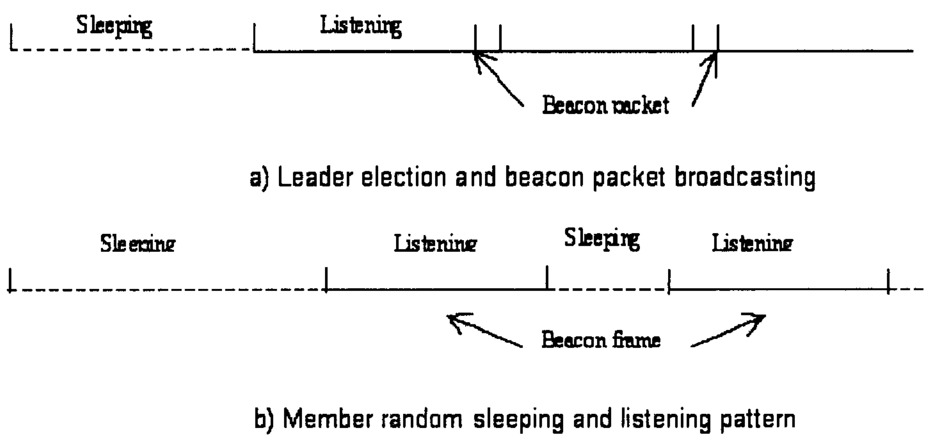

Sleeping
Listening
Beacon packet
a) Leader election and beacon packet broadcasting
Sleeping
Listening
Sleeping
Listening
Beacon frame
b) Member random sleeping and listening pattern

Fig. 7: Leader election. Cluster leaders are the sensors who wake up, at local, the earliest.

Since the leader broadcasts a beacon packet every beacon frame and every sensor's listening time period lasts exact the same time interval, the sensors within a leader's radio range would not miss that beacon packet. In other words, after a leader is elected and starts broadcasting beacons, the sensors in the leader's radio range would not elect itself to be a leader no matter when they wake up.

In the leader election phase, there are two important system parameters: the sleeping range T and beacon frame length F. Each node in the network will wake up at least once during T. At the end of the listening period F, every sensor becomes either a leader or a member. Thus, the leader election phase normally ends in $T + F$ time (With an exception that a small amount of extra time may be needed for solving the leader collisions).

The clusters formed by the leader election procedure present some interesting geometry and topology properties, which will be discussed in Section IV.3.

IV.2.2 The Beacon Frame

Although initially not synchronized, the sensors in a cluster are naturally synchronized to its leader beacon frame. We further divide each beacon frame into a number of time slots as shown in Fig. 8.

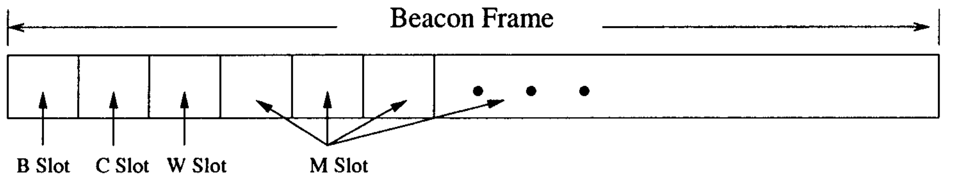


Fig. 8: Beacon frame for medium access control.

Each beacon frame starts and ends at every beacon packet sending time. The length of the beacon frame is predetermined as an application specific parameter. Since the slots in the beacon frame are dynamically assigned to the cluster members based on the sensing tasks, the number of slots is not related to the number of sensors in the cluster. The first slot of the beacon frame is called beacon slot (B slot) and is reserved for the cluster leader to send the beacon packet. The second slot is reserved for members to report leader collisions and is called collision slot (C slot). Since the leaders are elected at random, there is a small chance that neighbor leaders may elect themselves at the same time. A sensor can report the collision by sending a packet in the C slot. The third time slot (wakeup slot or W slot) is reserved for members to wake up leaders. To save energy, if it did not hear anything in its wakeup slot, each leader goes to sleep from the end of its wakeup slot until its next beacon sending time. The cluster members can wake up its leaders by sending a packet in the W slot. The rest of time slots of the beacon frame are called member slots (M slot). Cluster members use their member slots to send packets to the cluster leader. A sensor in a cluster acquires at most one member slot. Importantly, this mechanism endows the sensors that have acquired slots in the cluster with temporary ID numbers. Indeed, the slot number of an individual sensor becomes its temporary ID in the cluster. Upon receiving a beacon packet from the leader, a sensor can determine the followings:

- The next beacon sending time. (The length F of a beacon frame is known)
- The slot boundaries. (The length of a slot is known)

The leaders use a predetermined frequency channel to broadcast beacon packets in every beacon slot. Members use another special frequency channel to report leader collisions in the collision slot. In order to reduce power interference between neighbor clusters, each leader chooses a frequency channel at random from a relatively large pool. The chosen frequency is broadcast within the beacon packet and will be used by the cluster members to send packets to the leader. In this way, cluster members use their own cluster frequency channel to communicate with their leader. The channel will be different from their neighbor cluster frequency channels. Although leaders use a common channel to broadcast their beacon packets, leader election procedure and collision resolution mechanism guarantee that neighbor leaders (the leaders that their transmission ranges overlap with each other) send their packets at different times.

Members wake up their leaders by sending a packet in their leader's wakeup slot. If the leader receives the packet correctly (i.e. only one packet is sent at wakeup slot), the leader goes to sleep to save energy. However, if the leader only hears noise in its wakeup slot, it has to switch to listening mode (on its chosen frequency channel) during the rest of the beacon frame. Usually leaders will schedule the packet transmission and sending time. This wakeup mechanism is only needed when scheduling is not possible (such as when members are competing for member slots).

IV.2.3 Collision Resolution

Since the leaders elect themselves at random, it is possible that the beacon packets collide with each other. There are two situations that need the collision resolution mechanism: The actual beacon collision and the potential beacon collision.

The actual beacon collision refers to the situation where two or more nearby sensors wake up at exactly the same time (or very, very close) and decide to become leaders. The beacon packets sent by the new leaders are corrupted. The collided leaders may or may not have common members (the sensors that can hear the noise). If they do not have common members, due to the near/far problem, the new leaders have no way to detect the collision. But this causes no harm. If some sensor heard the collision, it sends a collision packet in the collision slot (starts at the time one slot late after the sensor switched from idle listening to receiving). If a leader heard any packet or noises in its collision slot, it randomly selects another time to send its beacon packet. Its beacon frame time period will start at that time. However, the leader will be in idle listening mode until it sends a new beacon packet. If it receives a beacon packet before it sends its own beacon packet (the collided leaders are neighbors), it will not send the beacon packet and become a member.

In the leader election phase, beacon packets sent by leaders are very short to avoid beacon collision. However, later on beacon packets may become much longer when data or other control information is inserted into beacon packets. If the beacon transmission times of two or more nearby leaders are so close that their beacon slots overlapped with each other, enlarged beacon packets may collide with each other. When a sensor receives some short beacon packets that are close enough to cause the collision problem, it sends a collision packet at the leaders' collision slot (C slot) and reset its listening timer (restart its listen time period). If a leader hears any thing at its collision slot, it randomly selects another time to send its beacon packet.

Note the sleeping interval $(0, T)$ contains a very large number for sensors to choose since the time unit is a micro-second (ms) or even a nano-second (ns). For instance, if $T = 1s$, the sleeping interval contains 10^6 (ms) or 10^9 (ns) numbers. Moreover, the beacon collision only happens at local. Hence, the chance of beacon collision is relatively small (Depends on the local density of the sensors).

The collision resolution mechanism ensures that neighbor leaders send beacon packets at separated times. It brings the system following advantages: (a) It reduces power interference. b). Neighbor leaders may use beacon packet sending time to identify themselves.

Occasionally, neighbor clusters may choose the same frequency -a frequency collision. The common members of nearby clusters (if any) may report to its leader and the leaders will randomly choose different frequencies –resolving the collision.

IV.2.4 Slot Competition

After leaders have been elected and clusters established, cluster members can compete for member slots in their beacon frame. First, a sensor randomly chooses a slot from the current "free" member slots in the beacon frame. Then the sensor sends a report packet that contains the chosen slot number to its leader. The report packet is sent twice (in wakeup slot and in the chosen slot). The leader will confirm the slot if it receives the packet. A frame bitmap describing its new frame usage will be sent within its next beacon packet. A sensor may select another free time slot if its previous chosen slot is not confirmed. For instance, if two packets are sent at the same slot, the leader will receive no packet at that slot (noises are heard and ignored by the leader).

The slot competition mechanism is designed as a building block that can be employed in various protocols. As it turns out, this mechanism provide a lightweight protocol for electing an active workforce, usually, a small subset of the local sensor population. (In Section IV.2.6). It is also used for the active gateway confirmation (In Section IV.2.5) and for the replacement leader selection (In Section IV.2.7).

IV.2.5 Active Gateway

In the network clustering phase, the sensors that wake up the earliest are elected to be cluster leaders. Each leader broadcasts a beacon packet in its beacon slot every beacon frame. A beacon packet contains a frequency channel number, a frame bitmap and an active gateway table. The frame bitmap describes the leader's frame usage. Each bit in the bitmap represents a slot in the beacon frame. 0 indicates a "free" slot. 1 indicates an "occupied" slot. An "occupied" slot represents a cluster member. The slot number is considered as the temporary ID of the cluster member.

Common sensors of two or more overlapped clusters are declared to be gateways. Adjacent clusters use their gateways to communicate with each other. In a dense network, the number of gateways of two clusters may be large. A leader can confirm one of its gateways as its active gateway for every neighbor leader. A leader has an active gateway table that stores its active gateway information. Each entry in an active gateway table contains a neighbor leader identifier (the beacon sending time) and a slot number, which is the temporary id of an active gateway that connected to that neighbor leader. Two overlapped clusters are connected if each cluster has at least one active gateway confirmed for the other cluster.

For example, consider adjacent leaders v and u in Fig. 9(a). The active gateway tables and beacon frames of these leaders are showed in Fig. 9(b) and (c). The active gateways connecting leader v and leader u are sensor A and B, respectively. Depending on the rule for choosing an active gateway, both sensor A and B can be the active gateway of leader u or v. Regardless of which sensor is chosen, a leader only chooses one sensor per neighbor leader as its active gateway.

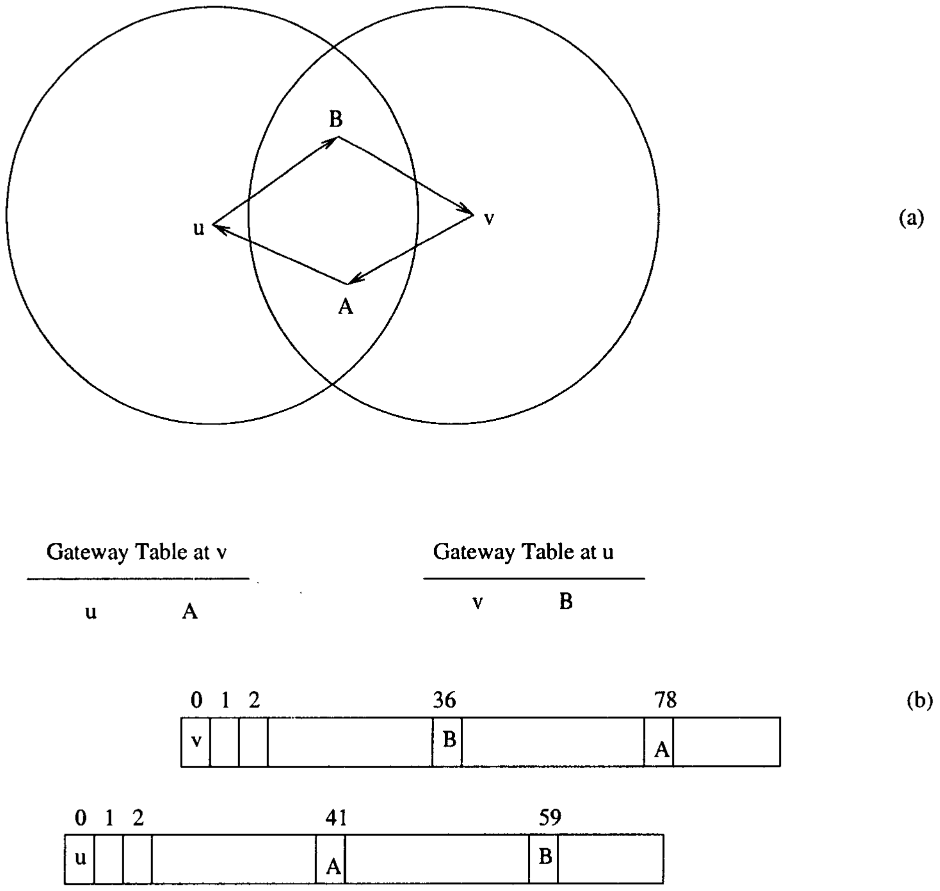


Fig. 9: Illustrating active gateways.

Each cluster member stores its leader's frame bitmap and gateway table. A gateway, which receives more than one beacon packet during its listening time period, stores a frame bitmap and a gateway table per leader. For example, Fig. 9(b) and (c) shows the leader frame information and active gateway tables that stored at gateway A and B. Using the gateway tables, a gateway can find out if two adjacent leaders are connected. The frame bitmap is used for a member to compete a "free" time slot as described in the previous section.

For instance, when gateway A receives a beacon packet from leader v, it stores the new frame information (the frame bitmap) and active gateway information (the gateway table) for leader v. Then it checks if leader v is connected to all its adjacent leaders (It checks the leaders active gateway table to find out if leader v has an active gateway for every adjacent leader). If an adjacent leader u is not in the gateway table of leader v, gateway A will randomly choose "free" slots for both leader v and u. Then it sends a report packet in each of those slots to leader v and u. Each report packet will contain the neighbor leader information (its beacon sending time and its chosen frequency channel). Based on the receiving report packets, a leader confirms one gateway per neighbor cluster. The leader may confirms the first reporting gateway as the active gateway and marks the corresponding bit as "occupied" in its bitmap. (The leader ignores the other gateways for the same neighbor leaders and leaves their chosen slots as "free"). The updated frame bitmap and active gateway table will be broadcast in its next beacon packet. Note that two neighbor cluster leaders may confirm different active gateways to each other, hence, there will be up to two active gateways between any pair of neighbor clusters. Also note that a sensor may become an active gateway of three or more neighbor clusters.

The active gateways are responsible for forwarding data between neighbor leaders. For instance, as showed in Fig. 9, if leader v wants to send data to leader u, it inserts the data into its beacon packet as payload and specifies that the payload is for leader u. The active gateway A forwards the payload to leader u in slot number 41 (in leader u's beacon frame and use leader u's frequency channel) and send an acknowledgement (ACK) to leader v in slot number 78 (in leader v's beacon frame and use leader v's frequency channel).

Once the leaders are elected and the gateways are confirmed, the infrastructure of the network is established. We refer the reader to Fig. 10 for an illustration.

From the energy saving point of view, let us consider the energy consumption in terms of the sensor awake time per beacon frame. A leader broadcasts a beacon packet in B slot and listens in C slot and W slot. Hence, given the number of slot F per beacon frame, a leader's awake percentage is $\frac{3}{F}$. Similarly, a cluster member listens for the beacon packet in each beacon frame. The awake percentage for a member is $\frac{1}{F}$. The awake percentage for an active gateway can be as much as twice or more.

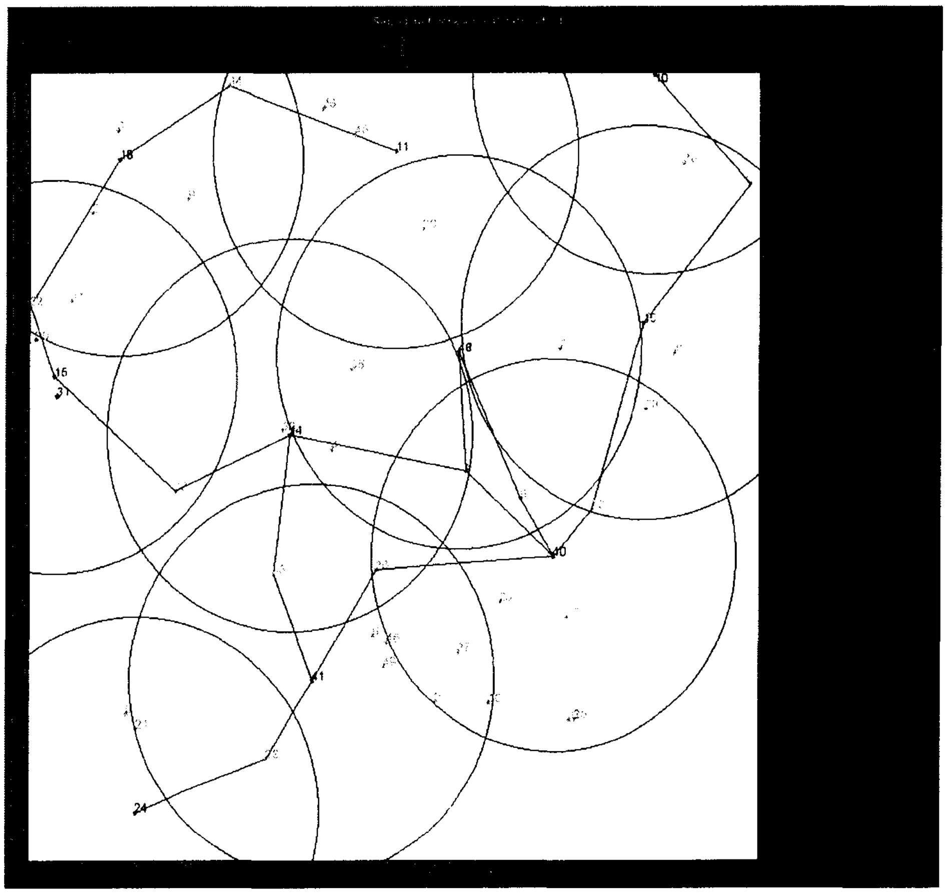


Fig. 10: An example of a sensor network communication infrastructure. The circles indicate clusters in the network. Cluster leaders are the dots located at the center of the circles. Two connected lines between two circles indicate a pair of connected clusters. The common end of these two lines is an active gateway.

IV.2.6 Active Workforce

As already mentioned once a sensor had acquired a slot, the offset of this slot in the frame provides a temporary ID that can be used in various protocols. This is a very important feature that can be exploited by a number of protocols that use the infrastructure provided. Indeed, a number of protocols in the WSN literature were designed with sensors with IDs in mind: such protocols do not work directly on anonymous networks. The fact that our clustering provides (locally unique) IDs allows these protocols to leverage our infrastructure with no additional overhead.

It is also worth noting that since the length F of the frame is a fixed system parameter and since the deployment of sensors is assumed to be massive, the number of sensors that acquire a slot in the cluster frame is usually a fraction of the total local sensor population. This choice was deliberate as it contributes to extending the longevity of the network. Also, this design decision is conducive of fault tolerance and security: should a sensor fail a fresh sensor is immediately available to take its place.

The active workforce of a cluster changes dynamically under the task executed. Indeed, in order to ensure longevity re-clustering may be necessary. In this context, an important question is to determine what sensors are eligible for participating in active workforce. The most obvious choice is to allow sensors to flip a coin in order to decide if they will take part in the competition for slots. Clearly, this strategy promotes load balancing and helps prolonging the longevity of the WSN.

The election of an active workforce is started by leader sending a beacon packet, which contains a competition probability p and an energy threshold E_t. Each member of the cluster first compares its remaining energy with the energy threshold, if its remaining energy is below the threshold, the member will not compete for a workforce. If its remaining energy is over the threshold, the member decides if competing for a workforce randomly based on the given probability.

Assume that the leader advertises k "free" slots in the beacon frame. If the leader lets its members compete for the workforce at "will", each cluster member chooses a slot at probability $\frac{1}{k}$. Then the expected number of elected workforce depends on the number of cluster members (N).

The probability that a slot is occupied by exact one sensor can be written as:

$$Pr[1] = \binom{N}{1} * \frac{1}{k} * (1 - \frac{1}{k})^{N-1} \tag{2}$$

Let X_1 be the random variable that counts the number of slots that are occupied by exact one sensor. The expected value of X_1 can be written as following:

$$\begin{aligned} E[X_1] &= k * Pr[1] \\ &= N * (1 - \frac{1}{k})^{N-1} \end{aligned} \tag{3}$$

Similarly, the probability that a slot is not occupied by any sensor can be written as:

$$Pr[0] = (1 - \frac{1}{k})^{N} \tag{4}$$

Table 2: Mean and standard deviation for X_1 (k=100)

N	100	200	300	400	500	600	700
$E(X_1)$	36.97	27.07	14.86	7.25	3.32	1.46	0.62
$\overline{X_1}$ $(n = 100)$	36.86	27.32	15.02	7.16	3.25	1.42	0.54
$S(X_1)$ $(n = 100)$	4.67	3.97	3.13	2.52	1.74	1.27	0.72
$\overline{X_1}$ $(n = 10)$	38.1	27.2	13.7	5.90	3.50	1.10	0.30
$S(X_1)$ $(n = 10)$	4.88	3.88	3.27	3.48	1.78	1.12	0.68

Table 3: Mean and standard deviation for X_0 (k=100)

N	100	200	300	400	500	600	700
$E(X_0)$	36.6	13.40	4.90	1.80	0.66	0.24	0.09
$\overline{X_0}$ $(n = 100)$	36.9	13.41	4.89	1.89	0.76	0.22	0.09
$S(X_0)$ $(n = 100)$	2.76	2.72	1.91	1.36	0.90	0.48	0.32
$\overline{X_0}$ $(n = 10)$	36.3	13.5	5.20	1.90	0.30	0.00	0.00
$S(X_0)$ $(n = 10)$	3.30	2.99	2.04	1.20	0.68	0.00	0.00

Let X_0 be the random variable that counts the number of slots that are not occupied by any sensor, the expected value of X_0 can be written as:

$$\begin{aligned} E[X_0] &= k * Pr[0] \\ &= k * (1 - \frac{1}{k})^N \end{aligned} \tag{5}$$

Table 2 and Table 3 gives the mean and the standard deviation of the samples for X_1 and X_0. We choose a large sample ($n = 100$) and a small sample ($n = 100$) to show the statistic properties of those random variables. The tables show that the means of the samples are very close to the expected values even for the small samples and the standard deviations of both samples are also small. In particular, the standard deviations of the samples for X_0 is smaller than that for X_1. It indicates the variance of X_0 is smaller, too.

Assume the leader needs to recruit a workforce of size m, we want to determine a probability (p) so that the expected value of X_1 is m:

$$E[X_1] = k * \binom{N}{1} * p * (1-p)^{N-1} = m \tag{6}$$

Thus, the solution of following equation will be the desired probability that used by cluster numbers to compete for slots.

$$f(p) = p * (1-p)^{N-1} - \frac{m}{N * k} = 0 \tag{7}$$

where the function $f(p) : [0,1] \rightarrow \mathbf{R}$

The derivative of function $f(p)$ is:

$$\begin{aligned} f'(p) &= (1-p)^{N-1} - p(N-1)(1-p)^{N-2} \\ &= (1-p)^{N-2}(1-p-pN+p) \\ &= (1-p)^{N-2}(1-pN) \end{aligned} \tag{8}$$

Since the derivative $f'(p)$ is a continuous function and $f(0) = f(1)$, Rolle's theorem guarantees that $\exists q, 0 < q < 1$, at which $f'(q) = 0$. Clearly, $q = \frac{1}{N}$ as showed in Fig. 11.

Notice that if $f(\frac{1}{N}) < 0$, no solution exists for the equation (7). If $f(\frac{1}{N}) \geq 0$, we can find a solution in $(0, \frac{1}{N}]$ by using Newton's method.

However, in order to solve the equation (7), the leader has to know the number N in advance. The number N can be estimated by the following procedure: the leader first "guesses" a probability, and let its member use this "guessed" probability to compete for slots. Then the leader estimates the number N based to the result of the slot competition.

Assume the slots that not occupied by any sensor is m_0, than based on equation (5), we have:

$$\begin{aligned} m_0 &\approx k * (1-p_0)^N \\ \log\frac{m_0}{k} &\approx \log(1-p_0)^N \end{aligned} \tag{9}$$

And the estimated value of N is:

$$N \approx \frac{\log\frac{m_0}{k}}{\log(1-p_0)} \tag{10}$$

The value of N can also be estimated based on equation (3). According to Table 2 and Table 3, however, we can get a better estimation of N based on equation (5). The estimation becomes very accurate after a few tries of the slot competition.

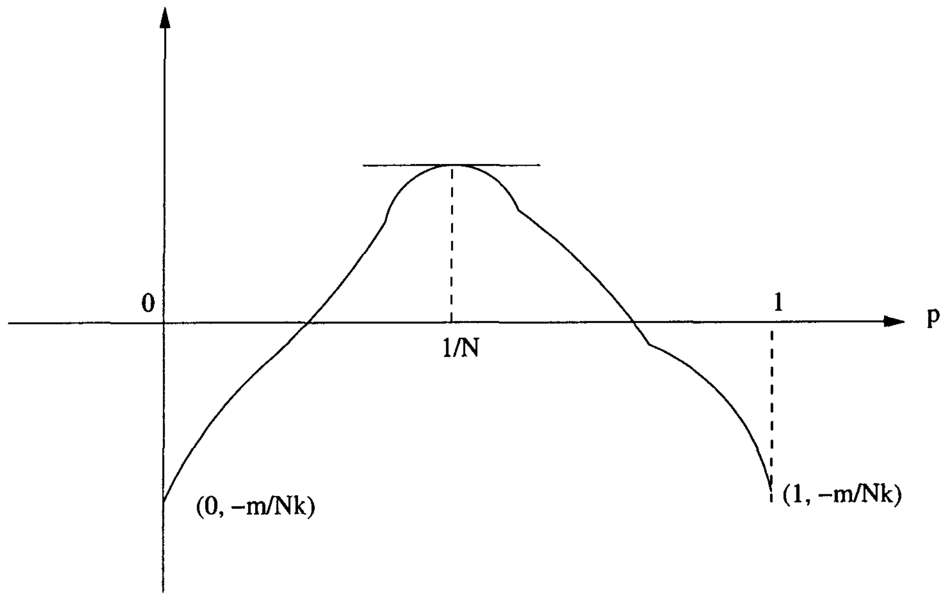


Fig. 11: Illustrating the function $f(p)$.

IV.2.7 Self-maintenance

An important functionality that the self-organization protocol needs to provide is to maintain the communication infrastructure. Sensors may fail due to many different reasons. When a leader and an active gateway dies, part of the infrastructure needs to be reorganized.

Re-clustering

In our protocol, leader failure can be aware immediately when its members miss a beacon packet. The members that are not gateways have to elect new leaders among themselves. (Gateways do not participate in leader re-election. However, active gateways will inform the neighbor clusters the leader failure). The leader election scheme is the same as described in IV.2.1. However, the sleeping range (backup range) can be much shorter since the number of sensors that participate in the leader re-election is smaller than before. When a new leader is elected, the gateway confirmation is also triggered to connect the new cluster into the system.

If a cluster member is inactive (i.e. is not involved in any sensing task or any communication task), it will wake up only at every beacon slot to receive leader's beacon packet. However, in order to maintain the communication infrastructure, every sensor (except leaders) needs to listen for a beacon frame once in a while. The reason is that old leaders may fail and new leaders may be elected.

Also, new sensors may be added to the system. It is inevitable that some new sensors may elect themselves as leaders since they belong to none of the old clusters. The members of old clusters must listen for a beacon frame once in a while in case new leader is elected. Since this is an energy consumption procedure, the interval between two listening is set to a very large value. We do not need this procedure to run fast since it just is a local problem that usually does not reduce the system performance dramatically.

Leader retirement

Cluster leaders play very important roles such as medium access controlling as well as sensing and task scheduling. Since cluster leaders normally consume more energy than cluster members, we developed a more aggressive maintaining scheme called *the leader retirement.* In order to provide continuous services and to avoid leaders becoming a single point of failure, a replacement leader can be selected before the original leader uses up its energy budget.

As showed in Fig. 12, when the remaining energy of a leader is below a given threshold (E_t), the leader starts a procedure for selecting a replacement leader among its members. Meantime, the leader continues to function until the replacement leader is selected. And then, the original leader is "retired" and the replacement leader will take over.

The criteria of a "good" replacement leader are the following:

1. Connectivity: The replacement leader should be within transmission range from all active gateways if possible;

2. Coverage: The replacement leader should be within transmission range from all the original cluster members if possible(leave no one behind);

3. Longevity: The replacement leader should have a large remaining energy budget.

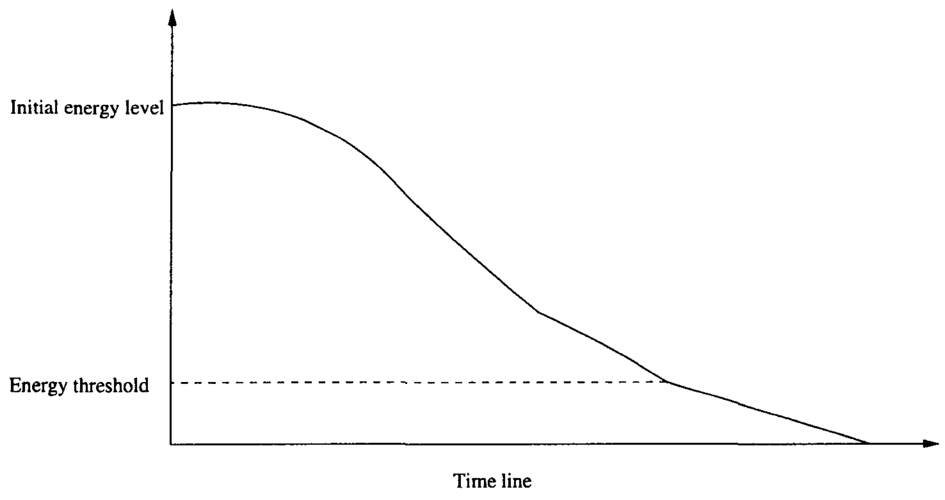


Fig. 12: Illustrating the energy consumption for a leader.

Fig. 13 shows the beacon frame and the frame bitmap during the replacement leader selection. Note that the member slots are divided into smaller slots. The "occupied" slots (indicated by 1 in the frame bitmap) reserves the space for active gateways, current workforce and neighbor leader beacon slots.

One way to choose a "good" replacement leader is to use "topology sensing" scheme proposed in [4]. The topology sensing scheme takes two beacon frame periods. In frame 1, each member (non gateways and its remaining energy is over the threshold E_t) in the cluster broadcasts a probe packet in a "free" slot and listens to all other slots. Each active gateway also broadcasts an active gateway packet in its reserved slots. Each member counts the number of probe packets it received and also counts the number of active gateway packets it received. After frame 1, each member will find out who are its neighbors. In frame 2, the leader broadcasts a new frame bitmap (Fig. 13(c)) that indicates which probe packet actually went through. Each member that secures a slot in frame 1 reports its counts and its remaining energy to the leader. After frame 2, the leader knows its cluster topology and the remaining power of its members. The leader is able to choose the "best" replacement leader among its members.

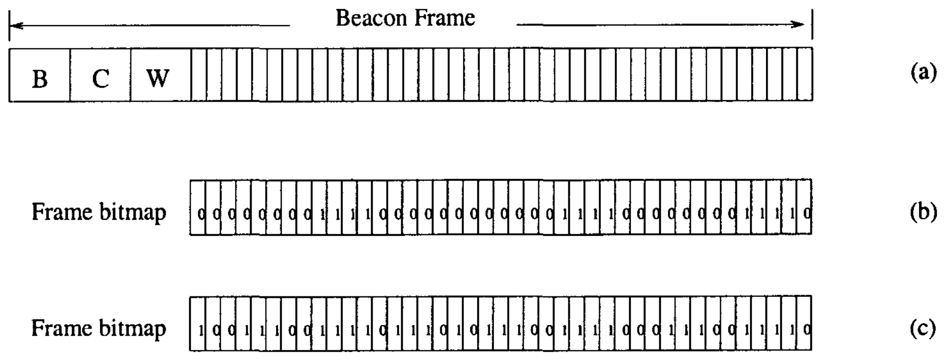


Fig. 13: Illustrating the beacon frame for replacement leader selection.

The original leader chooses a replacement leader based on member's connectivity, coverage and longevity (in that particular order). Then the leader announces its retirement and the new leader in its next beacon packet. The new leader starts its beacon frame at the beginning of its chosen slot and broadcast its beacon packets. The active gateways that can receive the new beacon packet remain as the active gateways (their temporary IDs are changed accordingly, though). The retired leader will become a member of the new cluster.

If some members can not receive the beacon packet from the new leader, they will elect a leader among themselves as described in previous section.

Active gateway retirement

Since an active gateway has to be awake in the beacon slots of all neighbor clusters to which it belongs, it consumes more energy than a normal cluster member. Similar to the leader retirement, a replacement active gateway can be selected before the energy budget of the current active gateway is used up.

The remaining question is what should be done in the case where two clusters are left without gateways. One of possible solutions is to establish a link between clusters by using "half-gateways". An example of the "half-gateways" is illustrated in Fig. 14. However, in order to find each other, those "half-gateways" have to send probe packets and monitor the radio channel to receive probe packets, which causes extra energy consumption.

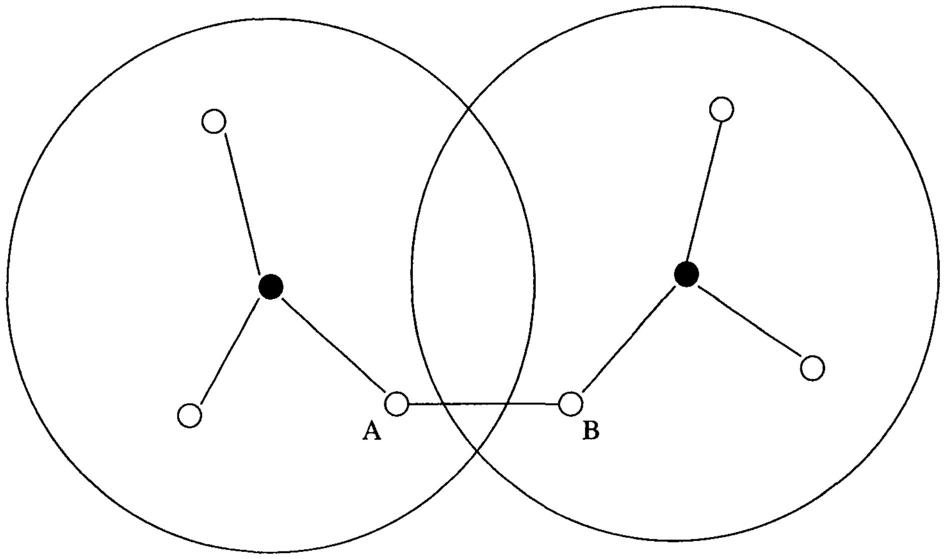


Fig. 14: Half-gateway example: node A and B are half-gateways.

IV.2.8 A New Look at the Beacon Frame

For simplicity, we divide the beacon frame into equal length slots. Each slot is sufficiently long to send a packet that contains the sensing data. However, the control packets are normally much shorter than the data packets. In previous section, we showed that the length of the slot can be changed to fit a particular requirement.

The collision packet for collision resolution is one of the control packets that are very short. Hence, we can combine the C slot and W slot into one slot as showed in Fig. 15. This slot is divided into many mini slots. The first mini slot is used by members sending collision packet. And we call it the C mini slot. The rest of the mini slots are used by members waking up the leader and are called W mini slots.

Using the new beacon frame, the leader's awake slots can be reduced to 2 per beacon frame. It is a significant saving in term of energy consumption. Furthermore, some W mini slots can be reserved for the active gateways so that the leader will know exactly when to wake up to listen to a packet sent by an active gateway.

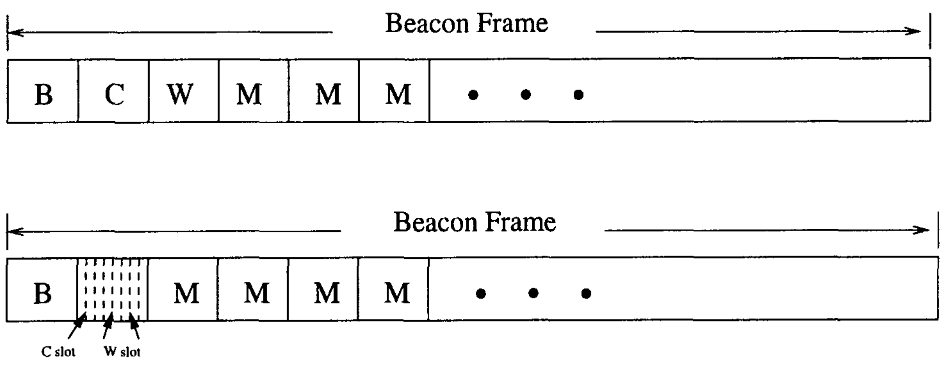


Fig. 15: Beacon frame for medium access control.

IV.3 TOPOLOGY PROPERTIES AND SIMULATION RESULTS

IV.3.1 Topology Properties of the Infrastructure

In this section, we present the properties of the infrastructure built by the self-organization protocol. A simplistic model of the infrastructure is based on the following assumptions:

- N sensors lie in a bounded area A^2. We assume the area is a square;
- The sensors are deployed randomly inside the bounding area;
- Each sensor has a transmission range of r. Hence, the cluster leaders in the network must be at least r apart (Assuming no leader collision).

Using the method developed in [69], a cluster is modeled as a circle of radius $\frac{r}{2}$, centered at the cluster leader. As illustrated in Fig. 16, these circles will never overlap due to the assumption that the cluster leaders are at least r apart. The problem of finding the maximum number of clusters can be restated as a packing problem: what is the maximum number of non-overlapping circles of radius $\frac{r}{2}$ that can be packed in a plane? It is proved in [69] that if the bounding area is given, the maximum number of clusters is fixed, and does no depend on the number of sensors in the network. The problem of finding the expected number of clusters can

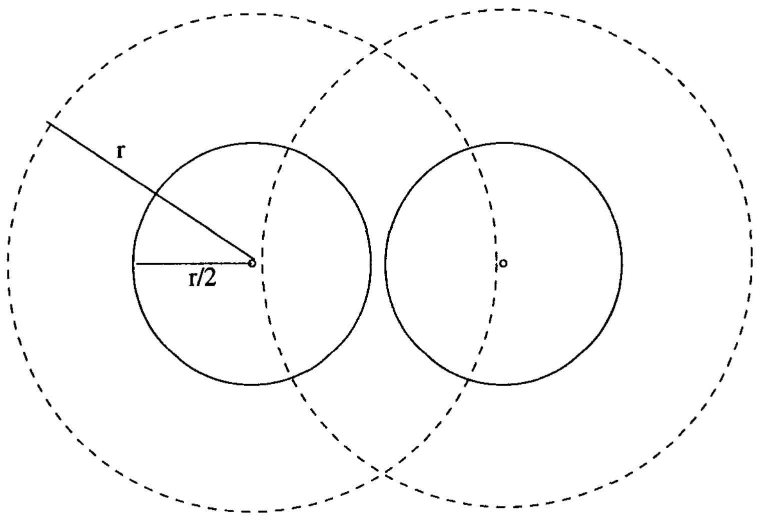


Fig. 16: A cluster is modeled by a circle of radius $\frac{r}{2}$.

be restated as a 2D parking problem [83, 85]: what is the expected number of circles of radius $\frac{r}{2}$ that are randomly placed on a surface such that none overlap and also no uncovered space large enough to fit another circle? At present, no analytical solution exists for the 2D parking problem. The computer simulation and the experimental results obtained in [25, 74] suggest that the expected cover limit is 0.547 for the 2D parking problem.

Fig. 17 shows the actual bounding area for the circles that model the clusters. Assume the inner box is the sensor bounding area, then the bounding area for the circle of radius $\frac{r}{2}$ is illustrated by the outer box. Using the value of the 2D parking limit, we have

Property IV.1 *The expected number $E(n)$ of clusters in a square of area A^2 can be approximately evaluated by the following equation:*

$$E(n) = \frac{0.547 * (A + \frac{r}{2})^2}{\pi(\frac{r}{2})^2} \tag{11}$$

Since neighbor cluster leaders must be at least r apart, the circles of radius $\frac{r}{2}$ that model the neighbor clusters must be fit within an annulus as showed in Fig. 18.

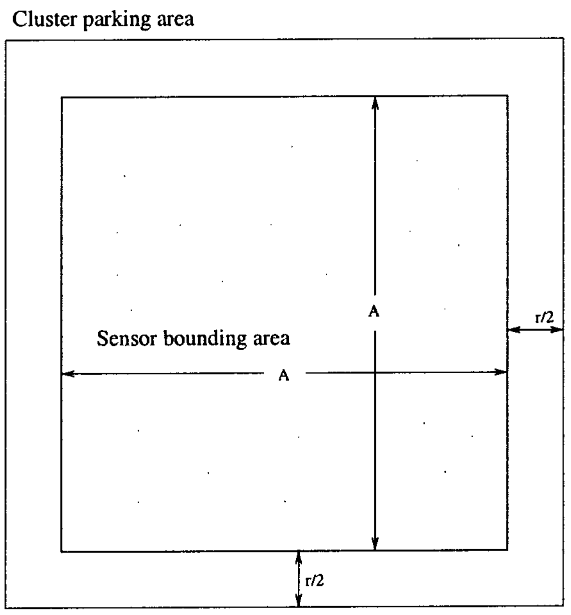


Fig. 17: Cluster parking area.

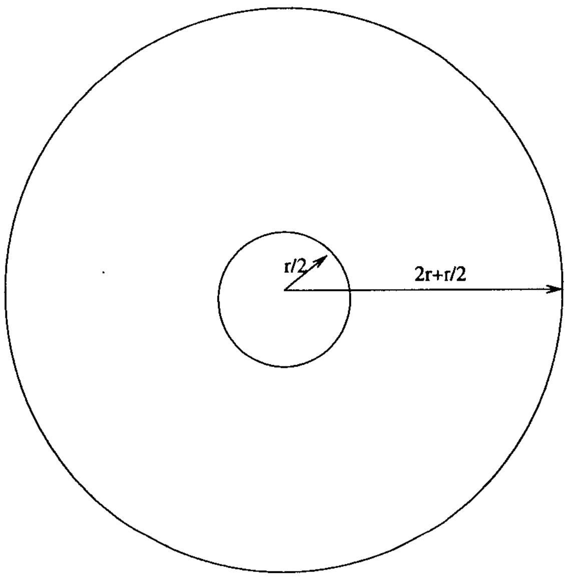


Fig. 18: Neighboring cluster packing area.

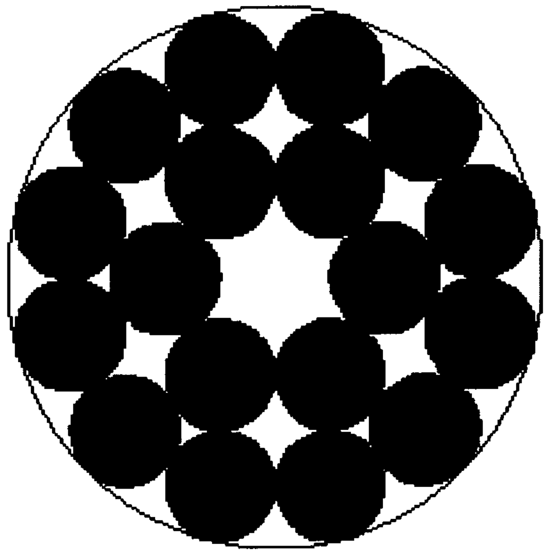


Fig. 19: 18 circles of radius $\frac{r}{2}$ are packed inside an annulus of inner radius $\frac{r}{2}$ and outer radius $2r + \frac{r}{2}$.

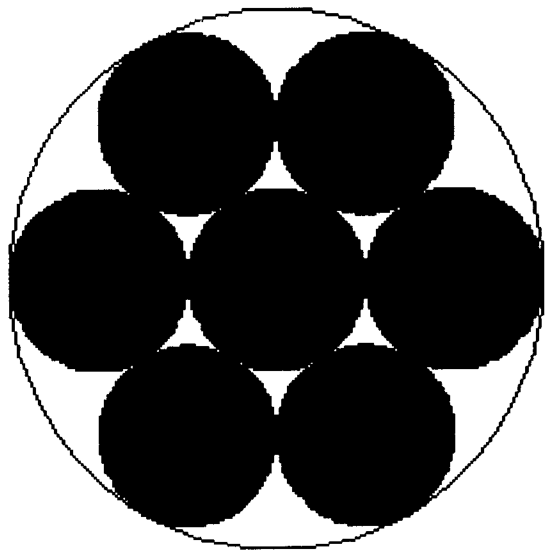


Fig. 20: 7 circles of radius $\frac{r}{2}$ are packed inside a circle of radius $\frac{3r}{2}$.

Using the result found by Kravitz in 1967 [22], 18 circles of radius $\frac{r}{2}$ can fit within the annulus (see Fig. 19). We claim that:

Property IV.2 *The degree of a cluster is at most 18.*

Note the area of the annulus in Fig. 18 is $6\pi r^2$, the expected degree of a cluster can be approximately evaluated as:

$$E(c) = \frac{0.547 * 6\pi r^2}{\pi(\frac{r}{2})^2} = 13.13 \tag{12}$$

Property IV.3 *The expected degree of a cluster is close to 13.13.*

Similarly, using the result that proved by Graham in 1968 [22] (see Fig. 20, we claim that:

Property IV.4 *A gateway can belong to no more than 7 clusters.*

And the expected number of clusters to which a gateway belongs can be approximately evaluated as:

$$E(g) = \frac{0.547 * \frac{9}{4}\pi r^2}{\pi(\frac{r}{2})^2} = 4.92 \tag{13}$$

Property IV.5 *The expected number of clusters to which a gateway can belong is close to 4.92.*

IV.3.2 Simulation Results

Our test-bed was implemented using PARSEC [78], a discrete-event simulator developed at UCLA, by placing the sensors randomly in a square of size 800mx800m. The given radio range is 200m and the beacon frame has length of 100 slots. The sensors wake up at random times, which are distributed uniformly over (0, T), where T is the sensor sleeping interval.

Constructing the infrastructure

We are interested in the performance of our self-organization algorithm and the performance of data delivering using the constructed communication infrastructure. In addition to implementing and simulating our own self-organization protocol (referred to as CMACS) we have also implemented and simulated the best-known self-organization protocol of Sohrabi et al. [96, 97, 98] that we refer to as SMACS.

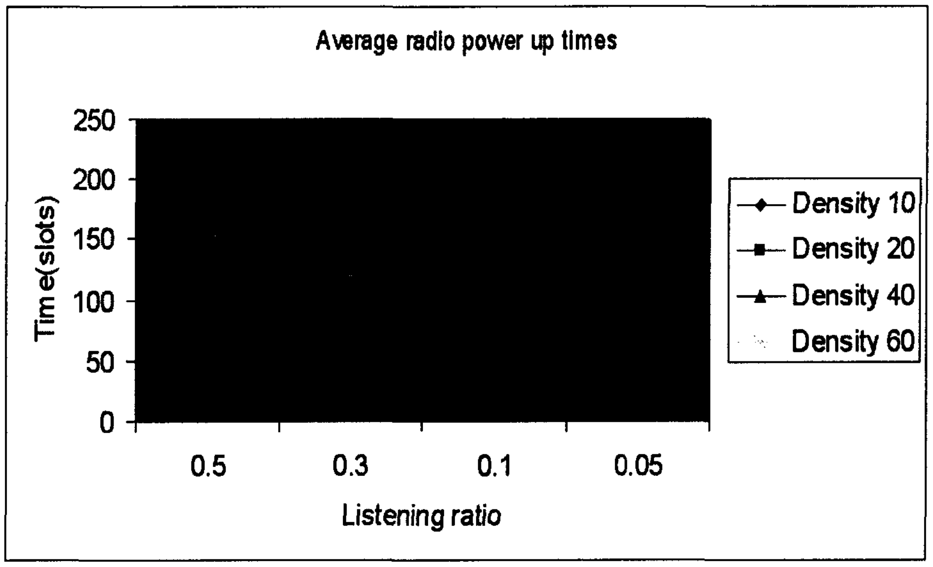


Fig. 21: Performance of the SMACS protocol (Energy consumption).

Among other things, we have compared and contrasted the performance of these two protocols in terms of the amount of energy and time required to complete the self-organization protocols. We use time slots to measure energy consumption, and have assumed that transmitting consume twice the amount of energy as listening or receiving.

In both protocols, energy consumption in network self-organization is greatly influenced by two factors: 1) the listening ratio at sensor level and 2) the network density at network level. Basically, sensor listening ratio specifies the average percentage time that the sensor is awake. In SMACS, the listening ratio is "bootup period: super frame" [96]. In CMACS, the listening ratio is "beacon frame: average sleep time (T/2) + beacon frame". Note that we keep the transmission range and area size constant in our simulation, thus, increasing network density merely means increasing the number of sensors in the network.

Fig. 21 and 22 demonstrate the performance of the SMACS protocol. When network density remains constant, average energy consumption decreases a little bit

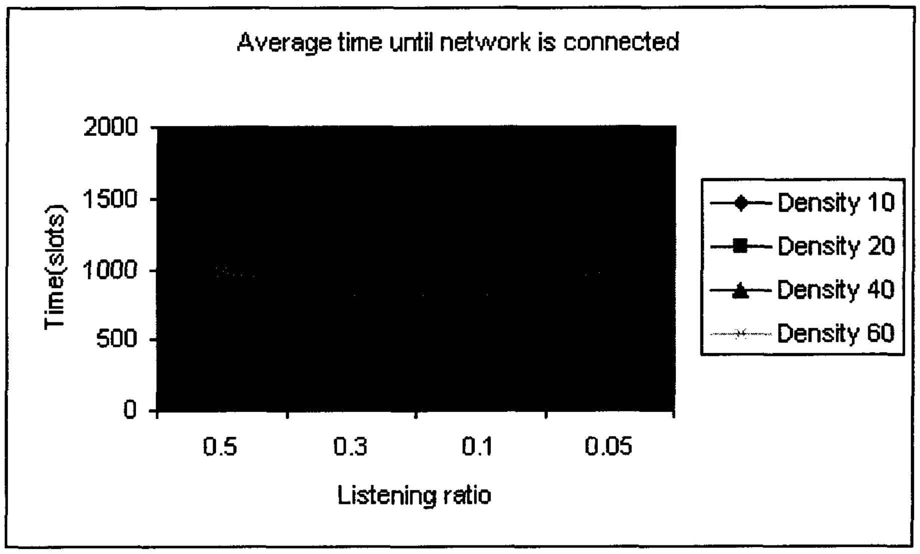


Fig. 22: Performance of the SMACS protocol (Connection time).

as listening ratio decreases. After the listening ratio reaches a certain point, the energy consumption starts to increase because the connection time increases rapidly. One can see this trend clearly when the network density is small (at 10 and 20). And when the listening ratio remains constant, average energy consumption and connection time decreases as network density increases. However, after the network density reaches a certain point, the energy consumption and connection time starts to increase.

Fig. 23 and 24 demonstrate the performance of our CMACS protocol. Similar to SMACS, after listening ratio reaches a certain point, the performance starts to get worse. And we get better performance when the network is dense because, as the network density increases, average energy consumption does not increase back up. The reason is that the number of backbone sensors (leaders and active gateways) in our protocol remains small as the network density increases (Fig. 25). Furthermore, the number of backbone sensors approaches a constant when the network density is large enough.

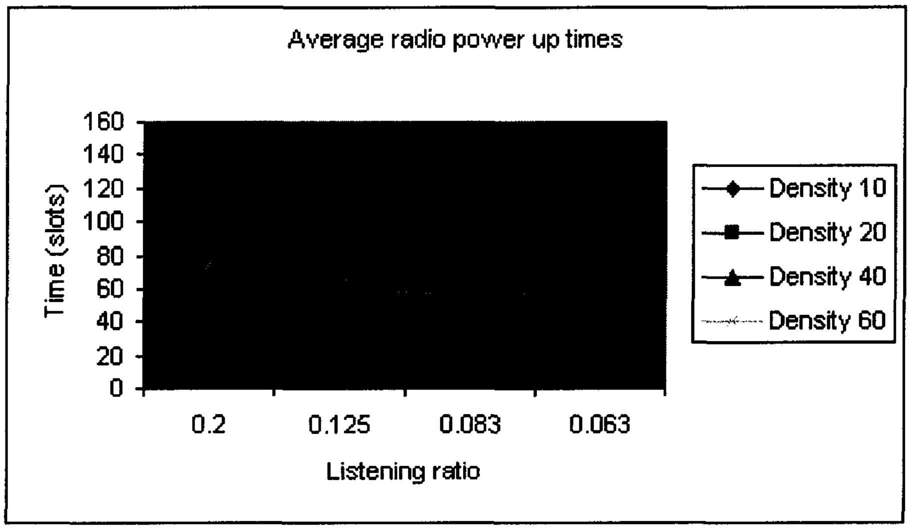


Fig. 23: Performance of the CMACS protocol (Energy consumption).

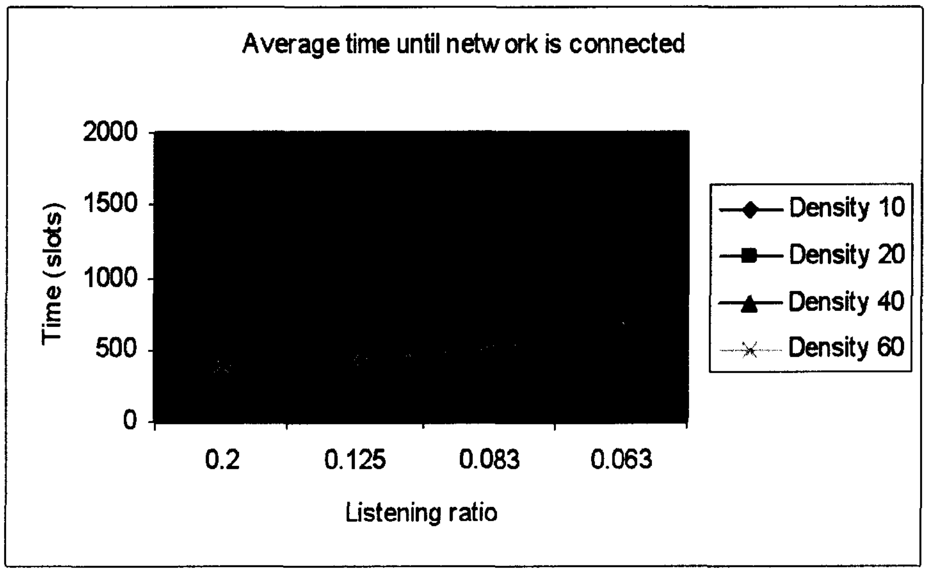


Fig. 24: Performance of the CMACS protocol (Connection time).

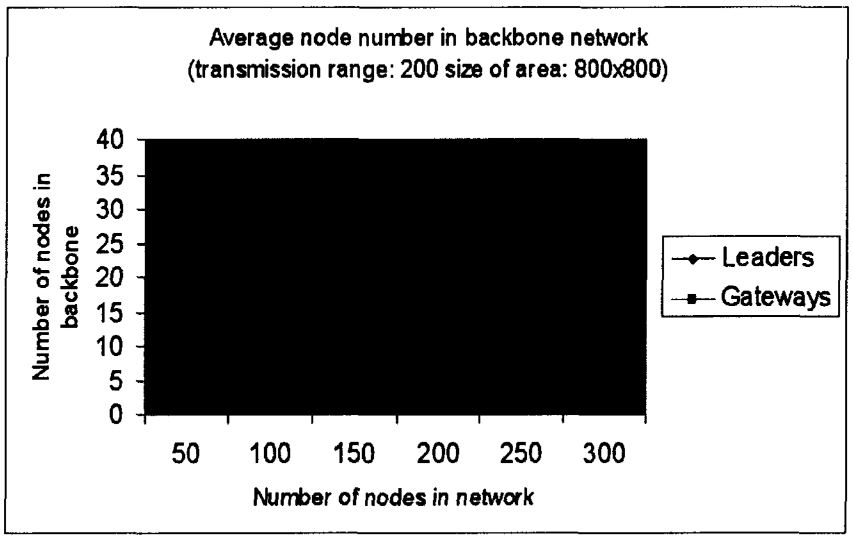


Fig. 25: Illustrating the average number of nodes in the backbone network.

A lesson learned from the simulation is that although intuitively we want the sensors to be turned off as long as possible, it is not always the best strategy for energy saving. If the listening ratio of sensors is too small, they may spend much more time to accomplish a certain task (e.g. self-organization) and eventually spend more energy than a scheme with a larger listening ratio.

Fig. 26 and 27 compare the best performance of CSMACS (listen ratio is 0.083) and SMACS (listen ratio is 0.1). CSMACS has better performances in terms of the amount of energy and time required to connect the network than SMACS does. The differences become larger when the network becomes denser. The reason is that in order to connect the whole network, CSMACS only needs to set up the backbone network. The other sensors in the network attached to the backbone automatically. And the number of sensors in the backbone network comparing to the number of overall sensors is very small, especially when the network is dense. The figures showed that both protocols are scalable to the number of overall sensors and CSMACS has better scalability to the network density than SMACS does.

Fig. 28 represents the real energy consumption ratios (wakeup ratio) for CSMACS and SMACS at different network density. In both protocols, the ratios are little over 0.10. It indicates that both protocols are energy efficient. Although we can decrease

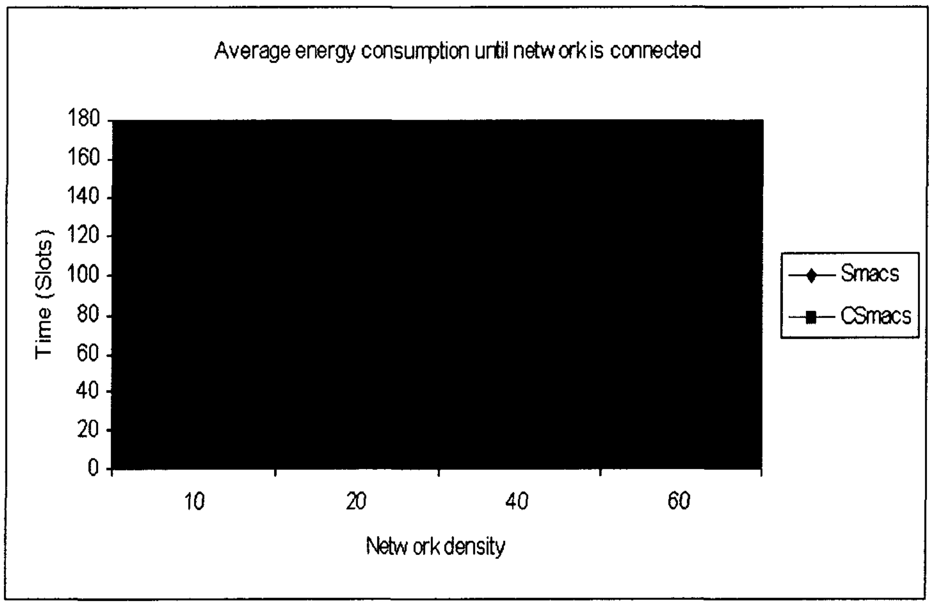


Fig. 26: Comparing the connection time.

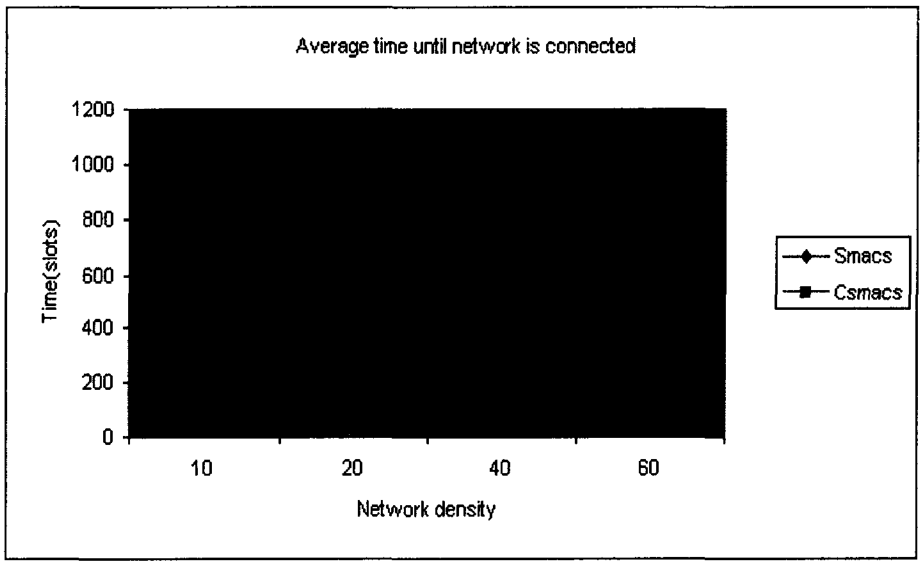


Fig. 27: Comparing the energy consumption.

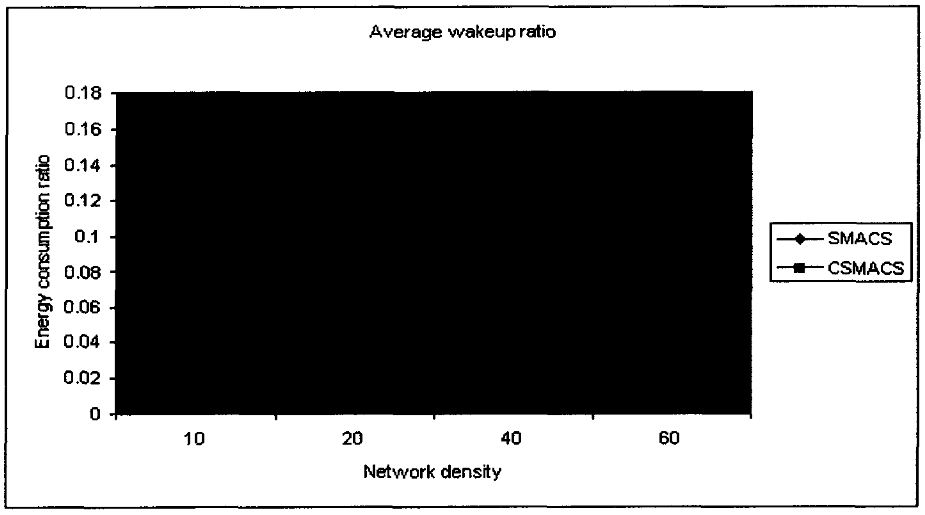


Fig. 28: Comparing the best wakeup ratio.

this ratio further by decreasing the listening ratio further. The performance in term of energy saving will get worse. This effect is showed in Fig. 21, 22, 23 and 24.

Flooding overhead and response time

The next two figures illustrate the performance of the communication infrastructure constructed by the two protocols plus the IEEE 802.11 MAC protocol as an extra reference in Fig. 29.

Fig. 29 shows the query message overhead of a simple flooding, in which query messages are forwarded only once. Using IEEE 802.11 protocol, every sensor in the network has to forward a query once. The number of message transmitted is the number of sensors in the network. SMACS does not take any advantage of broadcasting. In fact, it turns the broadcasting mechanism into many uni-castings. When a sensor receives a query, it has to forward it to every one of its neighbor sensors, except the sender. Hence, the flooding overhead is very high. In CMACS, only the backbone sensors (leaders and gateways) forward query messages. We get the least amount of query forwarding as a result. An interesting observation we had is that although sensors using SMACS protocol forward a query much more often than IEEE 802.11 protocol, they actually consume less energy. It is simply because

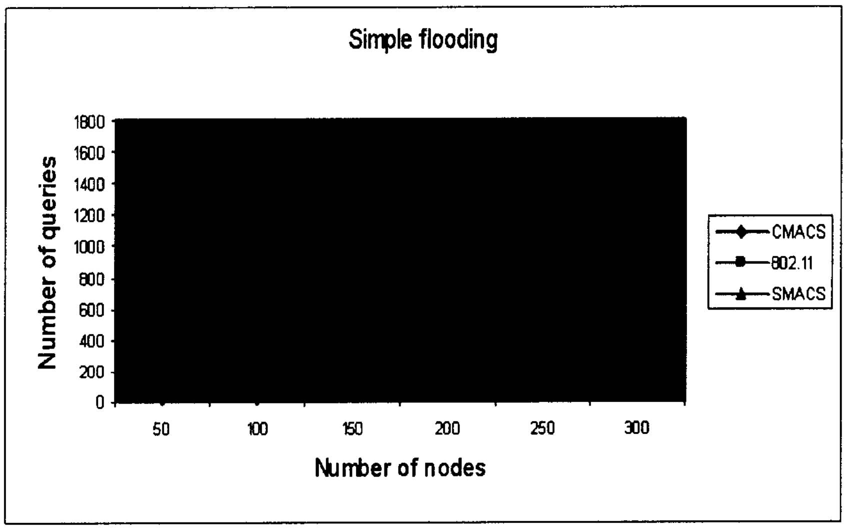


Fig. 29: Simple flooding overhead comparison.

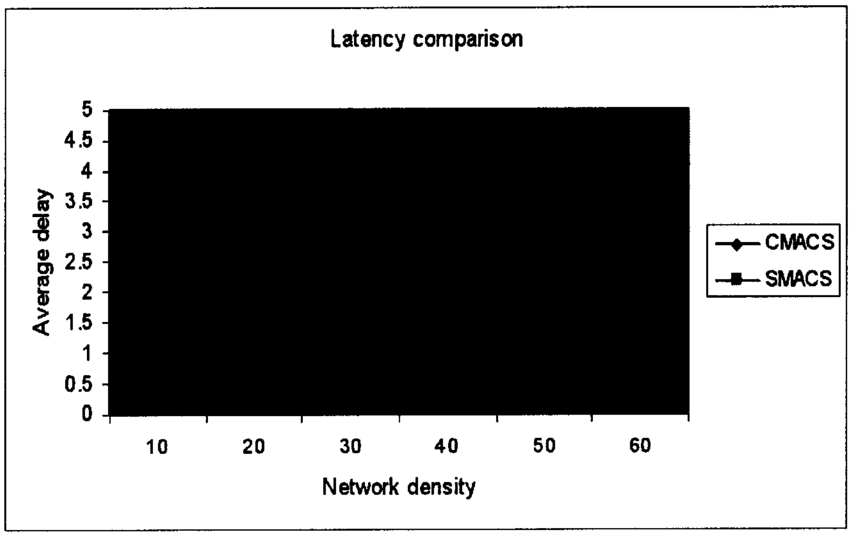


Fig. 30: Response time measured in time frame.

in IEEE 802.11 protocol sensors burn themselves in idle listening.

We also measure the average response time that indicates the latency observed between transmitting and receiving sensing data (Fig. 30). In both protocols, the length of the time frame (the beacon frame in CMACS and the super frame in SMACS) determines the response time. When this frame is longer, more energy is saved, but the response time is also longer.

Connectivity

The question of network connectivity is: How many neighbors should each node be connected to, so that the overall network then becomes connected? In other words, what should the network density be, so that the overall network then becomes connected? This problem was considered in a series of papers [31, 48, 70, 81, 103, 114] beginning in the 1970s and it is still open question by now. The best result in theory has been proved is that when network density $> N_0 = 10.526$ [81], the network is almost surely connected. In simulation, N_0 is suggested to be between 6 and 8 [70]. Previous section shows that the expected degree of a cluster is close to 13.13, which grantees the network connection in statistic sense. However, this expected degree of a cluster can only be reached when the density of the sensor network is close to infinity. Because of lacking of the analytical result, we investigated the network connectivity issue in our simulation.

Fig. 31 shows that when the network density is about 20, the connectivity will be close to 1. However, we found out that one kind of disconnection (one cluster is isolated from others) appears frequently and it is easy to fix. After this kind of disconnection is fixed, the connectivity will be close to 1 when the network density is 14.

IV.4 SUMMARY

In this chapter, we proposed a simple strictly localized self-organization protocol for massively deployed wireless sensor networks consisting of tint, anonymous sensors, very much like the Smart Dust sensors [45]. As a result of running the self-organization protocol a virtual infrastructure is obtained. One of the virtues of this infrastructure is the creation of a powerful multi-hop communication infrastructure capable of utilizing the limited resources of sensors in an adaptive and efficient way.

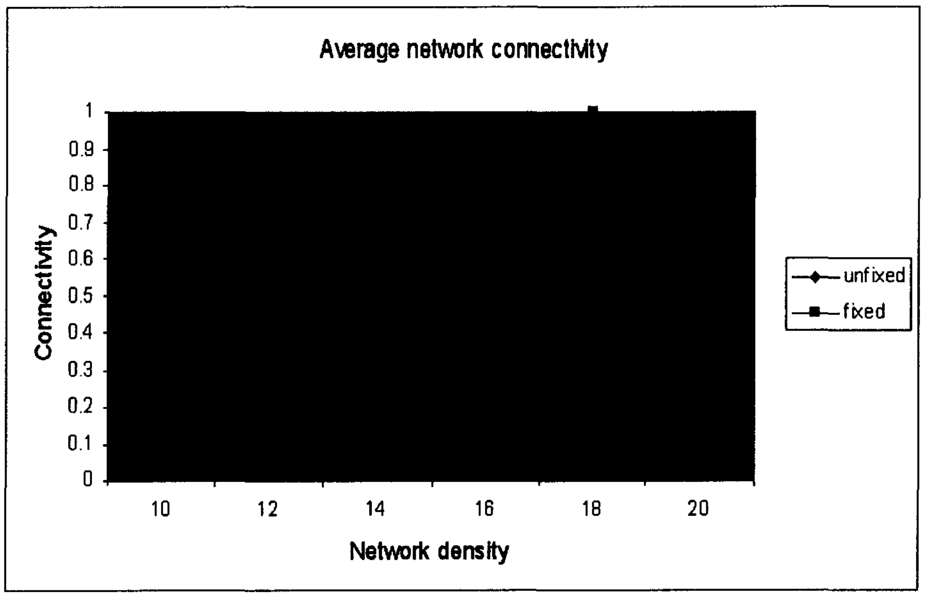


Fig. 31: Infrastructure connectivity.

In the following chapters, we will demonstrate that by leveraging this infrastructure one can design various efficient protocols for the sensor network. The sensor network is evolving. Resource-centric, network-centric, and data-centric architectures for WSN have been proposed. We envision that these architectures will converge toward a standards-based service-centric architecture analogous to the Web services architecture, albeit lightweight. Our current research leverages the proposed infrastructure for constructing such a service centric architecture for WSN. This promised to be a fascinating direction for future work.

CHAPTER V

TRAINING

V.1 INTRODUCTION

While some applications require sensory data with exact geographic location [54, 72], motivating the development of communication protocols that are location aware and perhaps location dependent, in most applications, exact geographic location is not necessary: all that individual sensors need is coarse-grain location awareness. There is, of course, an obvious trade-off: coarse-grain location awareness is lightweight but the resulting accuracy is only a rough approximation of the exact geographic coordinates.

The random deployment of sensors implies that the sensors are initially unaware of their exact location. Further, due to limitations in form factor, cost per unit and energy budget, individual sensors are not expected to be GPS-enabled; moreover, many probable application environments limit satellite access. It follows that the sensors have to learn either their exact geographic location (if specifically required by the application) or else a coarse-grain approximation of their location. The former task is referred to as *localization.* Wadaa et al. [108, 109] proposed to refer to the task of endowing individual sensors with coarse-grain location awareness as *training.*

Wadaa et al. [108, 109] proposed elegant training protocols for sensor networks. They obtained their training protocol by inducing the entire deployment area a dynamic coordinate system. They argued that the coordinate system provides a partitioning into clusters and a structured topology with natural communication paths. However, these protocols assume that the sink and the sensors are somehow synchronized. While a number of synchronization protocols have been published in the literature [20, 75, 93, 95], synchronization is an additional overhead that makes training somewhat complicated.

The main contribution of the work presented in this chapter is to propose a set of asynchronous training protocols for massively-deployed sensor networks. The sensors wake up according to their internal clock and are not engaging in a synchronization protocol with the sink. Our protocols are truly lightweight and simple to implement. We show analytically that in spite of the lack of synchronization, individual sensors are trained very efficiently. Extensive simulation results have confirmed that our asynchronous training protocols are energy efficient, each sensor being awake for a total time interval that is proportional with its wake-to-sleep ratio.

The remainder of this chapter is organized as follows: Section V.2 lays down the background of sensor network training and discusses the related work. Section V.3 and V.4 propose two novel training protocols. Section V.5 is the backbone of the entire chapter, presenting the theoretical underpinnings of various training processes. Section V.6 presents the simulation model and the simulation results. Finally, Section V.7 offers concluding remarks.

V.2 BACKGROUND

V.2.1 Sensor Coordinate System

Before we discuss training in detail, it is appropriate to give the reader more information about the type of sensor network assumed in this work. We assume an autonomous sensor network where a number of sinks provide local aggregation points for the data collected by sensors. Each sink organizes the sensors in a disk of radius R into an autonomous, short-lived sensor network as illustrated in Fig. 32.

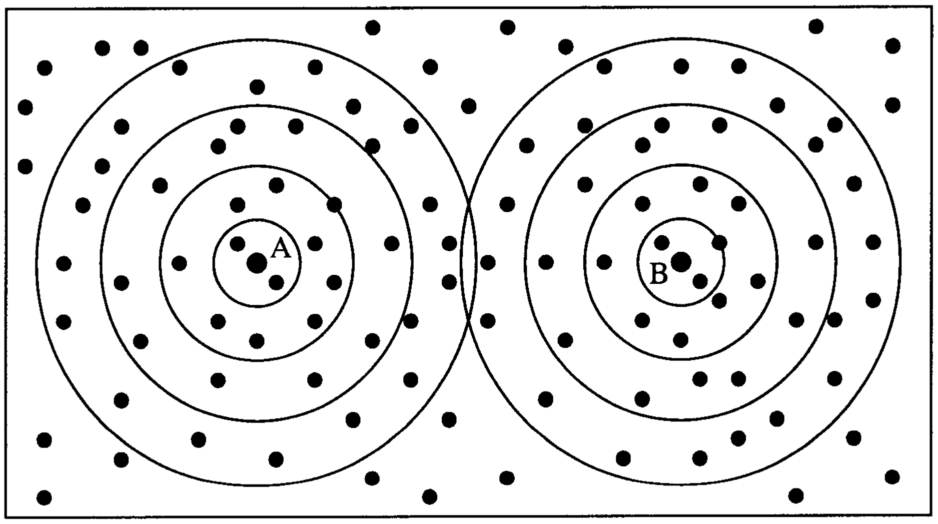


Fig. 32: Illustrating autonomous sensor networks, each trained by a sink.

For ease of understanding we shall present the training process centered at one of the several sinks in the deployment area. It is important to understand that each sink is engaging in the training process either independently of other sinks or in a concerted effort. Fig. 32 shows two sinks, each training independently the sensors in its neighborhood.

The task of training refers to imposing a coordinate system onto the sensor network in such a way that each sensor belongs to exactly one sector. The coordinate system divides the sensor network area into equiangular wedges. In turn, these wedges are divided into sectors by means of concentric circles or coronas centered at the sink and whose radii are determined to optimize the transmission efficiency of sensors-to-sink. Sensors in a given sector map to a cluster, the mapping between clusters and sectors is one-to-one. Referring to Fig. 33, the task of training a sensor network involves establishing:

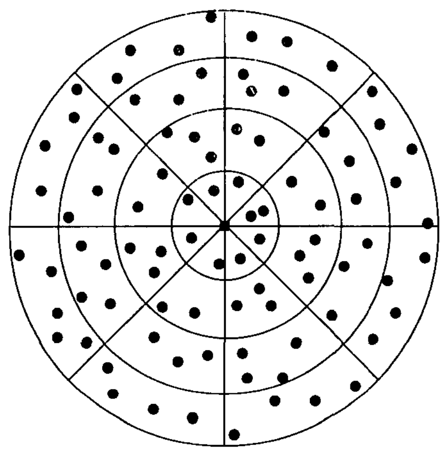


Fig. 33: A trained sensor network.

Coronas: The deployment area is covered by k coronas determined by k concentric circles of radii $r_1 < r_2 < \cdots < r_k$ centered at the sink.

Wedges: The deployment area is ruled into a number of angular wedges centered at the sink.

As illustrated in Fig. 33, at the end of the training period each sensor has acquired two coordinates: the identity of the corona in which it lies, as well as the identity of the wedge to which it belongs. Importantly, the locus of all the sensors that have the same coordinates determines a cluster.

In the remainder of this chapter, we discuss various protocols for the corona number training. The wedge number training is similar. The process of wedge number training can be done separately after the sensors learn their corona numbers. We are not going to discuss it in this chapter but refer to [109].

V.2.2 Sensor Cycle and Sink Cycle

The energy efficiency requirement demands the sensors be in the sleep mode most of the time. Hence, the sensor radios have to work on a low duty cycle, called a *sensor cycle*, in which the sensors will be awake only a short period of time (referred as *awake interval*) and sleep for the rest of the time.

Referring to Fig. 34, the sensor cycle is on length L (time units): the sensor is in sleep mode for $L-d$ and is awake for d. We assume the value of L is fixed during the network lifetime. However, the value of d and the beginning of the awake interval can be varied in different protocols.

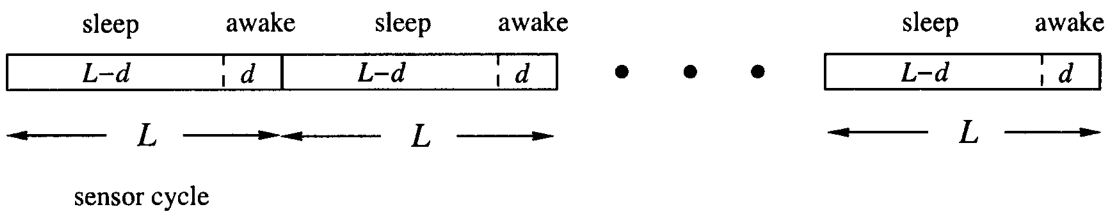


Fig. 34: Illustrating the sensor sleep-awake cycle.

In order to train the sensors in the network, the sink broadcasts training beacons. Assuming that k coronas $s_1, s_2, \ldots, s_k$ have to be established, the sink *transmission cycle* involves k broadcasts in each k – *cycle*. We assume the sink can transmit beacons at different power level. At the highest power level (P_k), the beacon can be received by the sensors in the outmost corona s_k, At the lowest power level(P_1), the beacon can only be received by the sensors in the corona s_1. The sink repeats the transmission cycle (referred as the *sink cycle*)in case that some sensors miss the previous training beacons. To distinguish between sink cycles and sensor cycles, we shall refer to sink cycles as $k - cycles$ and to the sensor cycles as $s - cycles$.

V.2.3 Synchronous Sensor Training

To make this dissertation self-contained, we now present the detail's of the training protocol of Wadaa at al. [108, 109]. Since the sensors are not aware of the beginning time of the training, the sink continuously repeats a call to training specifying the current time and a rendezvous time. As showed in Fig. 35, the sink repeatedly broadcasts a SYNC beacon at the highest power level for L time units. Each sensor can be only awake one time unit (slot) per $s - cycle$ during the synchronization to save energy. It guarantees that every sensor in the range receives the SYNC beacon at least one time in the synchronization phase.

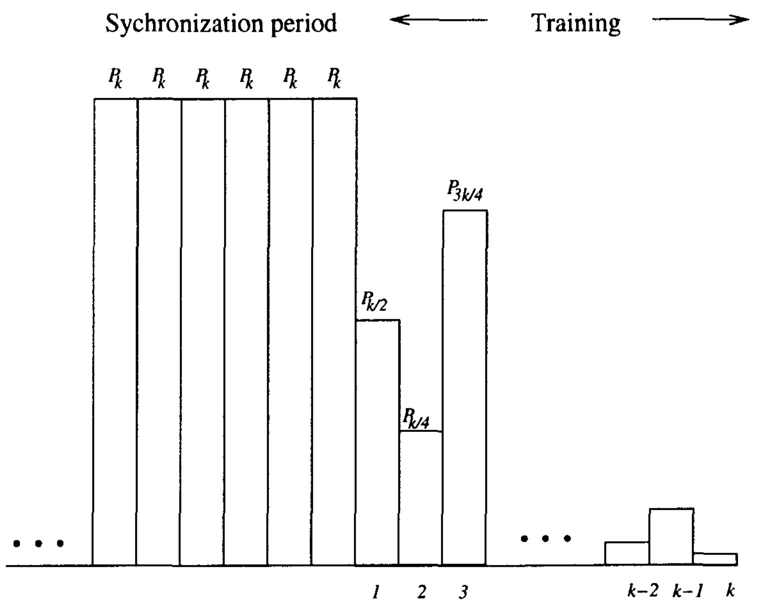


Fig. 35: Synchronization and training.

At the rendezvous time, the sink starts to transmit a TRAIN beacon with a power level corresponding to $P_{k/2}$. In other words, in the first training slot the sensors in the first $k/2$ coronas will receive the TRAIN beacon above a certain threshold, while the others will not. After the first training slot, the sensors are divided into two parts. The sink goes on transmitting at different power level, which divides the sensors into smaller parts until each part contains sensors that belong to one corona.

Assumes the value of L is known by the sink. The total time slot needed for sensor training is $L + k - 1$ and the sensor awake time slot number is $\log k + 1$, where the training awake time is $\log k$ and the synchronization awake time is 1.

The type of protocols described above is referred as the *synchronous training* protocol, which requires that sensors learn the beginning time of the training. The sensors can be trained without synchronization. In this type of protocols, which is referred as the *asynchronous training* protocol, the values of k and L are critical for energy saving. The protocol requires that the sensors know the value of k in advance in order to minimize energy consumption.

V.3 THE ASYNCHRONOUS TRAINING PROTOCOL

V.3.1 The Protocol

The main goal of this section is to propose an efficient asynchronous training protocol for a massively-deployed collection of sensors endowed with a centrally-placed sink.

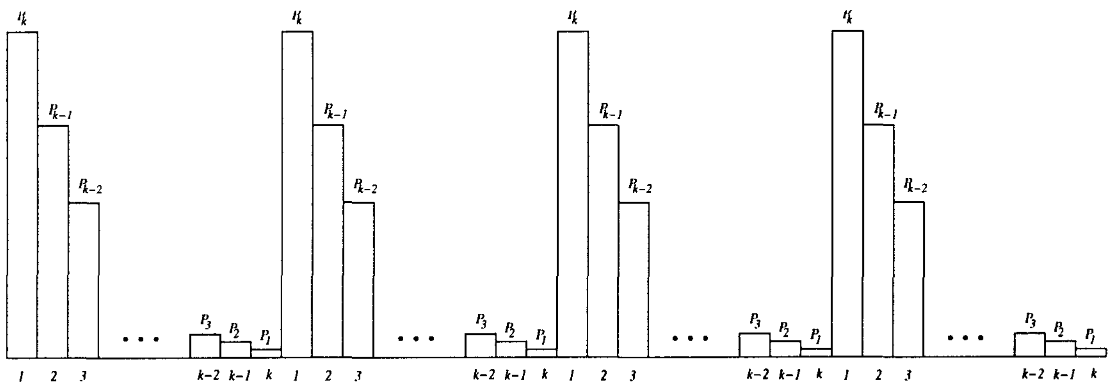


Fig. 36: Illustrating the sink transmission cycles.

The idea of the protocol is as follows: The sink repeats the transmission cycle illustrated in Fig. 36 (without synchronization phase). Assuming that k coronas $s_1, s_2, \ldots, s_k$ have to be established, the sink *transmission cycle* involves k broadcasts at successively lower power levels. The sink starts out by transmitting at the highest power, sufficient to reach the sensors in the outmost corona s_k; next, the sink transmits at a power level that can be received in corona s_{k-1} but not s_k. This is, then, continued until in the sink transmits at a power level that can be received only by the sensors in corona s_1.

The sensors alternate between sleep and awake mode in each $s-cycle$. During the awake period, the sensors determine their corona number with the following rules:

A sensor determines that it belongs to corona s_1 by receiving a P_1 beacon. Or the sensor determines that it belongs to corona s_i (where $i \neq 1$) by:

1. Receiving a P_{s_i} beacon

2. Failing to receive a $P_{s_{i-1}}$ beacon (which need not be in the same $k - cycles$.)

The way the $k - cycle$ is set up, in slot $i (1 \leq i \leq k)$ the sink transmitting at a power level P_{s_i}, where $s_i = k - i + 1$.

In the next section, we will show analytically that the sensors be aware of the value of k is critical for protocol efficiency.

V.3.2 Selecting Optimal Sensor Parameters

Recall that each sensor alternates between sleep periods and awake periods. Referring to Fig. 34, the sensor sleep-awake cycle is on length L: the sensor is in sleep mode for $L - d$ time and is awake for d time units.

The goal is to guarantee that the sensor is corona-trained within n ($n \geq 1$) $s - cycles$ – *regardless* of the moment when it wakes up for the first time.

The main goal of this subsection is to derive analytically formulae that allow to tailor both L and d (as functions of n and k) in such a way as to achieve this goal as efficiently as possible.

To begin, observe that since a sensor is awake for d time units per $s - cycle$, the smallest value of d that guarantees that the sensor can be trained in n $s - cycles$ is

$$d = \left\lceil \frac{k}{n} \right\rceil. \tag{14}$$

To simplify the notation, in the remainder of this work we shall assume that $\frac{k}{n}$ is an integer. Assume, without loss of generality, that the first $k - cycle$ begins at time

0 and that for some arbitrary x, the sensor wakes up x time units after the beginning of the $k-cycle$. With this assumption, it is easy to confirm that the n awake periods $A_1, A_2, \ldots A_n$ of the sensor are given for $(0 \leq i \leq n-1)$ by

$$A_i = [x + iL, x + iL + d]. \tag{15}$$

By projecting each of these intervals onto the generic interval $[0, k]$ of length k we obtain a new sequence $\overline{A_1}, \overline{A_2}, \ldots \overline{A_n}$ of intervals such that for $(0 \leq i \leq n-1)$

$$\overline{A_i} = [(x + iL) \bmod k, (x + iL = d) \bmod k]. \tag{16}$$

We note that the $\overline{A_i}$'s are *modulo* intervals in the sense that there may be a jump from the end of the generic interval to its beginning. Nonetheless, as we shall see this phenomenon does not create any difficulties.

In order to optimize coverage, we insist that the rightmost endpoint of the modulo interval $\overline{A_i}$ coincide with the left endpoint of the modulo interval $\overline{A_{i+1}}$. In other words, we are interested in determining L in such a way that

$$(x + iL + d) \bmod k = (x + (i+1)L) \bmod k. \tag{17}$$

In turn, (17) implies that

$$L - d \equiv 0(\mathrm{mod} k)$$

or, equivalently,

$$L = mk + d \tag{18}$$

for some natural number m.

Now, replacing the value of L suggested by (18) into (16) we obtain a new form for the modulo intervals $\overline{A_i}$. Specifically, we can write

$$\overline{A_i} = [(x + id) \bmod k, (x + (i+1)d) \bmod k]. \tag{19}$$

It is also easy to confirm that by virtue of (17) the union of the $\overline{A_i}$s covers the generic interval $[0, k]$ of length k. Thus, we have proved the following important result.

Theorem V.1 *Given that $d = \frac{k}{n}$, a sufficient condition for training a sensor in n s-cycles regardless of the first wakeup time is that $L = mk + \frac{k}{n}$ for some arbitrary natural number m.*

To help the reader visualize what is going on, suppose that $n = 3$, and refer to Fig. 37. The goal is to train the sensor in 3 $s - cycles-$ regardless of its first time x when it wakes up. Fig. 37(a) illustrates the $s - cycle$ as well as the wake-up periods in heavy lines. Fig. 37(b) shows that the projections of $\overline{A_1}$, $\overline{A_2}$ and $\overline{A_3}$ cover the interval $[0, k]$.

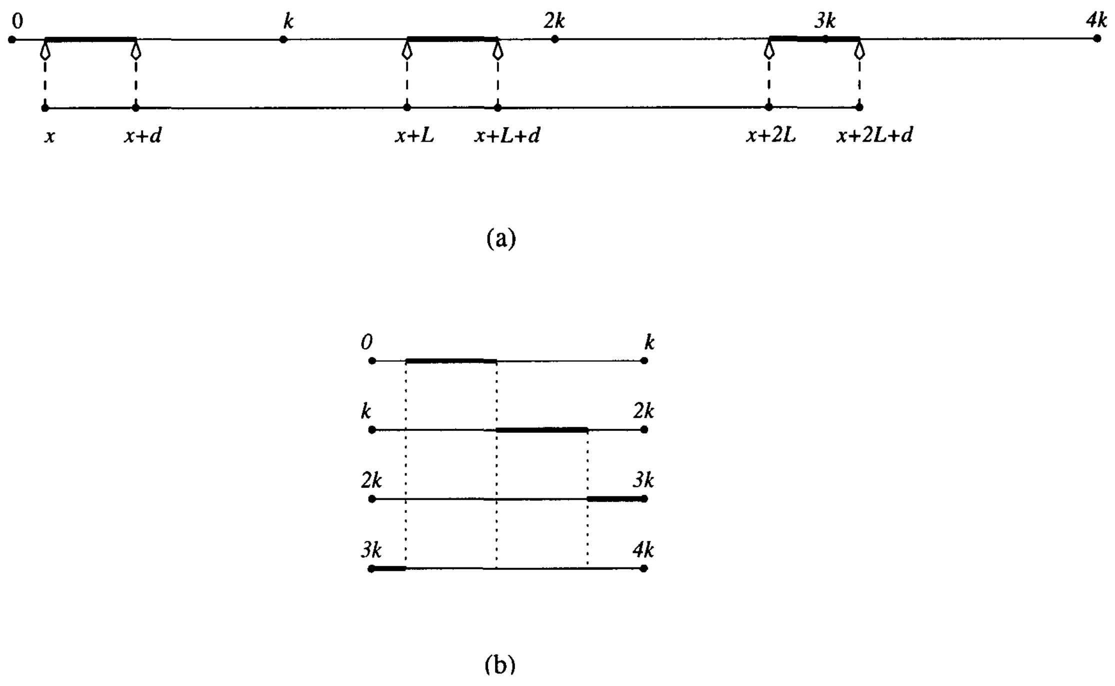


Fig. 37: Illustrating Theorem V.1 for $n = 3$.

V.4 HYBRID TRAINING PROTOCOLS

In this section, we propose a hybrid training protocol which combines the synchronization and training procedure. In the new protocol, synchronization and training are combined into one scheme. The sink sends two beacons in each slot instead of one. The first beacon (referred as *SYNC beacon* is sent with full power for sensor synchronization. It contains two numbers: the total corona number (k) and the current corona number (s). The second beacon (referred as *TRAIN beacon* is sent at successively lower power levels (outside in) for sensor training. Each TRAIN beacon contains the current corona number (s). The sink cycle is illustrated in Fig. 38.

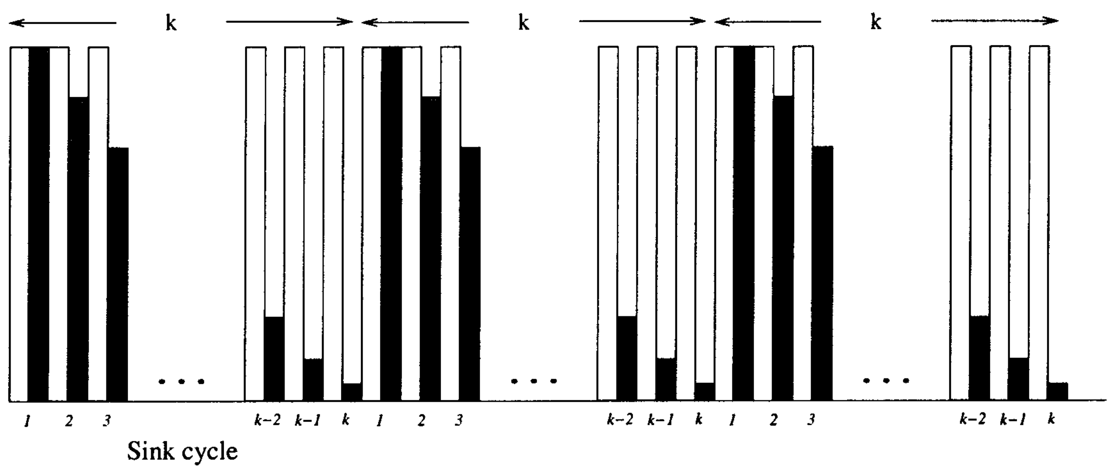


Fig. 38: Illustrating the hybrid training sink cycles.

For now we assume that a sensor is awake for one slot in every $k - cycle$. Before the training begins, sensors wake up at random until they receive a SYNC beacon. Thus, the probability of waking up in slot j is $\frac{1}{k}$. By receiving the beacon in a given slot $j (1 \leq j \leq k)$, the sensor determines the followings:

- Slot boundaries
- Synchronization to the $k - cycles$
- The value of k
- The beginning and end of the $k - cycle$

Consider a sensor that wakes up at random to learn its corona number i, for some $1 \leq i \leq k$. Assume that the sensor wakes up in slot j of some $k - cycle$ as showed in Fig. 39.

If the sensor receives a SYNC beacon but not a TRAIN beacon in slot j, the sensor knows its corona number is in range of $[s + 1, k]$. If the sensor receives a TRAIN beacon in the same slot, it knows its corona number is in range of $[1, s]$. In

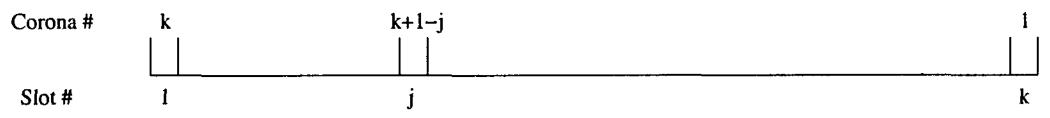


Fig. 39: Slot number and corona number.

order to figure out its corona number, the sensor can employ a linear search scheme or a binary search scheme on its corona number range.

In the linear search scheme, the sensor starts to listen from the highest boundary of its range until it misses a TRAIN beacon, which indicates its corona number is contained in the last TRAIN beacon that the sensor received. Or the sensor may reach the end of its range without missing any beacons, which indicates its corona number is the lowest boundary of its range. The reason for searching in descending order is because a sensor can learn its corona number immediately.

In the binary search scheme, the sensor starts to listen in the middle of its range and computes its new range based on whether the TRAIN beacon is received or not. Assume its corona number range is $[low, high]$, the sensor starts to listen at slot $\lceil\frac{low+high}{2}\rceil$, if the TRAIN beacon is received, its new range is $[low, \lceil\frac{low+high}{2}\rceil]$. Otherwise its new range is $[\lceil\frac{low+high}{2}\rceil + 1, high]$. The procedure continues until its range boundaries are equal. The boundary gives the sensor's corona number. The binary search scheme introduced here slightly differs from the traditional binary search algorithm. First, the scheme starts at an arbitrary position instead of the middle of the range. Second, the scheme ends when the searching range length reaches 1 instead of finding a particular value.

Note in slot j the sink is transmitting to the intention of corona s, where $s = k - j + 1$. If $i \leq k - j + 1$ the sensor receives the training beacon. It then goes to sleep and to wake up in slot $\lceil\frac{k-j}{2}\rceil$ of the next $k - cycle$. Clearly, the corona being "addressed" in slot $\lceil\frac{k-j}{2}\rceil$ is:

$$\begin{aligned} k - \left\lceil\frac{k-j}{2}\right\rceil + 1 &= k - \left\lfloor\frac{k-j}{2}\right\rfloor - \left\lceil\frac{k-j}{2}\right\rceil + \left\lfloor\frac{k-j}{2}\right\rfloor + 1 \\ &= k - (k-j) + \left\lfloor\frac{k-j}{2}\right\rfloor + 1 \end{aligned}$$

$$\begin{aligned} &= j + \left\lfloor \frac{k-j}{2} \right\rfloor + 1 \\ &= \left\lfloor \frac{k+j}{2} \right\rfloor + 1 \\ &= \left\lceil \frac{k+j}{2} \right\rceil \qquad (20) \end{aligned}$$

This corresponds to the intuition idea that the sensor proceeds by binary search in the range $[j+1, k]$.

On the other hand, if $i > k+1-j$, then the sensor does not receive the TRAIN beacon. The sensor goes to sleep and will wake up again in slot $\lceil \frac{j}{2} \rceil$ of the next $k - cycle$. It is easy to confirm that the corona addressed in this slot is:

$$\begin{aligned} k - \left\lceil \frac{j}{2} \right\rceil + 1 &= k - \left\lfloor \frac{j}{2} \right\rfloor - \left\lceil \frac{j}{2} \right\rceil + \left\lfloor \frac{j}{2} \right\rfloor + 1 \\ &= (k-j) + \left\lfloor \frac{j}{2} \right\rfloor + 1 \\ &= \left\lfloor \frac{2k-j}{2} \right\rfloor + 1 \\ &= \left\lceil \frac{2k-j+1}{2} \right\rceil \\ &= \left\lceil \frac{k+(k-j+1)}{2} \right\rceil \qquad (21) \end{aligned}$$

V.5 ANALYSIS

V.5.1 Hybrid Training with Linear Position Search

Table 4 shows the corona number range after the first SYNC beacon is received by a sensor. The table presents a k by k matrix, referred as *range matrix*, in which the row number i is the corona number that the sensor is actually in and the column number s is the corona number that obtained in the first received SYNC beacon. The matrix is the same for both the linear and binary searching schemes. The range matrix can be divided into two parts, the lower triangle $(i > s)$ gives the corona number range $[s+1, k]$ when the TRAIN beacon is not received. The upper triangle including the diagonal $(i \leq s)$ gives the range $[1, s]$ when the TRAIN beacon is received.

Let $C(i, s)$ be a function that evaluates the slots needed to train a sensor in corona i. The $C(i, s)$ is not equal to the length of corona number range because a sensor need not listen for the whole range in some case. Table 5, referred as *linear searching matrix*, shows the awake slots needed to train a sensor (after the first SYNC beacon is received) in a linear searching scheme. $C(i, s)$ is a function of the sensor's actual

Table 4: Searching range matrix

$i\backslash s$	1	2	3	$\cdots$	$k-2$	$k-1$	k
1	$[1,1]$	$[1,2]$	$[1,3]$	$\cdots$	$[1,k-2]$	$[1,k-1]$	$[1,k]$
2	$[2,k]$	$[1,2]$	$[1,3]$	$\cdots$	$[1,k-2]$	$[1,k-1]$	$[1,k]$
3	$[2,k]$	$[3,k]$	$[1,3]$	$\cdots$	$[1,k-2]$	$[1,k-1]$	$[1,k]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-2$	$[2,k]$	$[3,k]$	$[4,k]$	$\cdots$	$[1,k-2]$	$[1,k-1]$	$[1,k]$
$k-1$	$[2,k]$	$[3,k]$	$[4,k]$	$\cdots$	$[k-1,k]$	$[1,k-1]$	$[1,k]$
k	$[2,k]$	$[3,k]$	$[4,k]$	$\cdots$	$[k-1,k]$	$[k,k]$	$[1,k]$

Table 5: Searching awake time matrix

$i\backslash s$	1	2	3	$\cdots$	$k-2$	$k-1$	k
1	$s-i$	$s-i$	$s-i$	$\cdots$	$s-i$	$s-i$	$s-i$
2	$k-i+1$	$s-i+1$	$s-i+1$	$\cdots$	$s-i+1$	$s-i+1$	$s-i+1$
3	$k-i+2$	$k-i+1$	$s-i+1$	$\cdots$	$s-i+1$	$s-i+1$	$s-i+1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-2$	$k-i+2$	$k-i+2$	$k-i+2$	$\cdots$	$s-i+1$	$s-i+1$	$s-i+1$
$k-1$	$k-i+2$	$k-i+2$	$k-i+2$	$\cdots$	$k-i+1$	$s-i+1$	$s-i+1$
k	$k-i+2$	$k-i+2$	$k-i+2$	$\cdots$	$k-i+2$	$k-i+1$	$s-i+1$

corona number (i) and the first received corona number (s). In general, when the TRAIN beacon is received ($i \leq s$), the awake time period is $[i-1, s-1]$. When the TRAIN beacon is not received ($i > s$), the awake time period is $[i-1, k]$. Hence the awake time for searching is $s-i+1$ and $k-i+2$, respectively. This general rule has two exceptions: first, when $i = 1$, the awake time period is $[1, s-1]$ because the sensor knows its corona number if it receives a TRAIN beacon in which $s = 1$. Second, when $i = s+1$, the awake time period also does not include $i-1 = s$ because that is the slot which the sensor receives its first SYNC beacon (without the TRAIN beacon).

From the above observations, we can write $C(i, s)$ as the following. Note that

$C(i, s)$ is the searching slot number plus 1 (for receiving the first SYNC beacon).

$$\text{if } i = 1 \text{ then } C(1, s) = s - i + 1 \tag{22}$$

$$\text{else if } i = s + 1 \text{ then } C(i, s) = k - i + 2 \tag{23}$$

$$\text{else if } i > s + 1 \text{ then } C(i, s) = k - i + 3 \tag{24}$$

$$\text{else if } i < s + 1 \text{ then } C(i, s) = s - i + 2 \tag{25}$$

Recall the sensor's wakeup time is a random number. A sensor's wakeup time can be any number for 1 to k. The expected awake slot number for training a sensor in corona $i = 1$ is:

$$E(C_1) = \frac{k(k+1)/2}{k} \tag{26}$$

Observing the Table 5, in each row, the item number is $i - 2$, where $i > s + 1$, and 1, where $i = s + 1$, and $k - i + 1$, where $i < s + 1$. Hence, the expected awake slot number for training a sensor in a corona $i \neq 1$ is:

$$E(C_i) = \frac{(i-2)(k-i+3) + (k-i+2) + \sum_{s=i}^{k}[s-i+2]}{k} \tag{27}$$

Assume N sensors are uniformly distributed in the sinks transmission range R. In corona i, the expected sensor number is

$$E(N_i) = \frac{i^2\pi - (i-1)^2\pi}{k^2\pi}N = \frac{2i-1}{k^2}N$$

We can calculate the overall expected awake slot number for training using the following equation:

$$\begin{aligned} E(C) &= \frac{E(C_1)N/k^2 + \sum_{i=2}^{k} E(C_i)(2i-1)N/k^2}{N} \\ &= \frac{(k+1)k/2}{k^3} + \frac{\sum_{i=2}^{k}[(i-2)(k-i+3) + (k-i+2)] * (2i-1)}{k^3} + \\ &\quad \frac{\sum_{i=2}^{k}[(k-i+1)(k-i+4)/2] * (2i-1)}{k^3} \end{aligned} \tag{28}$$

By simplifying equation (28), we have

$$E(C) = \frac{k}{4} + 2\frac{1}{6} - \frac{5}{4k} - \frac{7}{6k^2} + \frac{1}{k^3} \tag{29}$$

In the worst case, the awake slot number for searching is $k - 1$, hence, the overall slot number T needed for training all the sensors is:

$$T = L + k - 1 \tag{30}$$

V.5.2 Hybrid Training with Binary Position Search

Definition V.2 (Searching distance) *Assume the corona number range is* $[low, high]$*, the searching distance (SD) is equal to* $high - low + 1$*.*

In a binary searching scheme, the searching awake time is closely related to the search distance defined by **Definition** V.2. We can transform the range matrix into a *distance matrix* as showed in Table 6:

Table 6: Searching distance matrix

$i \backslash s$	1	2	3	$\cdots$	$k-2$	$k-1$	k
1	s	s	s	$\cdots$	s	s	s
2	$k-s$	s	s	$\cdots$	s	s	s
3	$k-s$	$k-s$	s	$\cdots$	s	s	s
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-2$	$k-s$	$k-s$	$k-s$	$\cdots$	s	s	s
$k-1$	$k-s$	$k-s$	$k-s$	$\cdots$	$k-s$	s	s
k	$k-s$	$k-s$	$k-s$	$\cdots$	$k-s$	$k-s$	s

From the distance matrix, we can get the following function that computes SD.

$$\text{if } i \leq s \text{ then } SD = s \tag{31}$$

$$\text{else if } i > s \text{ then } SD = k - s \tag{32}$$

Definition V.3 (Binary position search) *For a given search distance* SD ($SD \in \mathbb{N}$)*and a position* i *in the range* $[1, SD]$*, the binary position search algorithm recursively divides the range into two small ranges, the search ends when a range contains position* i *only.*

By the definition, the binary position search times can be evaluated using the function $f(SD, i)$:

$$\text{where } f(1,1) = 0, (\text{recursive end condition})$$

$$f(SD,i) = 1 + f(\lceil SD/2 \rceil, i) \text{ if } 1 \leq i \leq \lceil SD/2 \rceil$$

$$f(SD,i) = 1 + f(SD - \lceil SD/2 \rceil, i - \lceil SD/2 \rceil) \text{ if } \lceil SD/2 \rceil < i \leq SD$$

Property V.4 $\lfloor \log SD \rfloor \le f(SD, i) \le \lceil \log SD \rceil$

Property V.5 *Give a search distance SD, the searching times $f(SD, i)$ is either $\lfloor \log SD \rfloor$ or $\lceil \log SD \rceil$ for any searching position i. The total number of position is equal to SD. And the number of position of which $f(SD, i) = \lfloor \log SD \rfloor$ is $2^{\lceil \log SD \rceil} - SD$. The number of position of which $f(SD, i) = \lceil \log SD \rceil$ is $2 * SD - 2^{\lceil \log SD \rceil}$*

Let $A(i)$ be a random variables that counts that slot number needed (on the average) to train a sensor in a fixed corona i, we can write:

$$A(i) = \frac{\sum_{s=i}^{k}[f(s,i)+1] + \sum_{s=1}^{i-1}[f(k-s, i-s)+1]}{k} \tag{33}$$

According to Property V.4, we have:

$$\underline{A}(i) \le A(i) \le \overline{A}(i) \tag{34}$$

Where

$$\underline{A}(i) = \frac{\sum_{s=i}^{k}(\lfloor \log s \rfloor + 1) + \sum_{s=1}^{i-1}(\lfloor \log (k-s) \rfloor + 1)}{k}$$

and

$$\overline{A}(i) = \frac{\sum_{s=i}^{k}(\lceil \log s \rceil + 1) + \sum_{s=1}^{i-1}(\lceil \log (k-s) \rceil + 1)}{k}$$

Since $s = k - j + 1$, we have:

$$\overline{A}(i) = \frac{\sum_{j=1}^{k-i+1}(1 + \lceil \log (k-j+1) \rceil) + \sum_{j=k-i+2}^{k}(1 + \lceil \log (j-1) \rceil)}{k} \tag{35}$$

$\overline{A}(i)$ can be evaluated as following:

$$\begin{aligned}
\overline{A}(i) &= 1 + \frac{\sum_{j=1}^{k-i+1}\lceil \log \underbrace{(k-j+1)}_{\alpha} \rceil + \sum_{j=k-i+2}^{k}\lceil \log (j-1) \rceil}{k} \\
&= 1 + \frac{\sum_{\alpha=i}^{k}\lceil \log \alpha \rceil + \sum_{\alpha=k-i+2}^{k}\lceil \log (\alpha - 1) \rceil}{k} \\
&= 1 + \frac{\sum_{\alpha=1}^{k}\lceil \log \alpha \rceil - \sum_{\alpha=1}^{i-1}\lceil \log \alpha \rceil}{k} \\
&\quad + \frac{\sum_{\alpha=1}^{k}\lceil \log \alpha \rceil - \sum_{\alpha=1}^{k-i}\lceil \log \alpha \rceil - \lceil \log k \rceil}{k} \\
&= 1 + \frac{2\sum_{\alpha=1}^{k}\lceil \log \alpha \rceil - (\sum_{\alpha=1}^{i-1}\lceil \log \alpha \rceil + \sum_{\alpha=1}^{k-i}\lceil \log \alpha \rceil + \lceil \log k \rceil)}{k}
\end{aligned} \tag{36}$$

Let C be the random variable describing the number slot number needed to train a sensor in the system, assuming that the total number of corona is k, the expected value $E[C]$ that we want to determine can be written as:

$$E[\underline{C}] \le E[C] = \frac{\sum_{i=1}^{k} A(i)(2i-1)}{k^2} \le E[\overline{C}] \tag{37}$$

Where

$$E[\underline{C}] = \frac{\sum_{i=1}^{k} \underline{A}(i)(2i-1)}{k^2}$$
$$E[\overline{C}] = \frac{\sum_{i=1}^{k} \overline{A}(i)(2i-1)}{k^2}$$

Then we have:

$$E[\overline{C}] =$$
$$\frac{\sum_{i=1}^{k}[1 + \frac{2}{k}\sum_{\alpha=1}^{k}\lceil\log\alpha\rceil - \frac{1}{k}(\sum_{\alpha=1}^{i-1}\lceil\log\alpha\rceil + \sum_{\alpha=1}^{k-i}\lceil\log\alpha\rceil + \lceil\log k\rceil)](2i-1)}{k^2}$$
$$= 1 + \frac{2}{k}\sum_{\alpha=1}^{k}\lceil\log\alpha\rceil - \frac{1}{k^3}\sum_{i=1}^{k}(2i-1)[\sum_{\alpha=1}^{i-1}\lceil\log\alpha\rceil + \sum_{\alpha=1}^{k-i}\lceil\log\alpha\rceil + \lceil\log k\rceil]$$
$$= 1 + \frac{2}{k}\sum_{\alpha=1}^{k}\lceil\log\alpha\rceil - \frac{1}{k}\lceil\log k\rceil - \frac{1}{k^3}\sum_{i=1}^{k}(2i-1)[\sum_{\alpha=1}^{i-1}\lceil\log\alpha\rceil + \sum_{\alpha=1}^{k-i}\lceil\log\alpha\rceil]$$

Our evaluation of $E[\overline{C}]$ relies on the following summations:

Claim 1 $\sum_{i=1}^{k}\sum_{j=1}^{i}\lceil\log j\rceil = (k+1)\sum_{i=1}^{k}\lceil\log i\rceil - \sum_{i=1}^{k} i\lceil\log i\rceil$

Proof.

$$\sum_{i=1}^{k}\sum_{j=1}^{i}\lceil\log j\rceil = \sum_{i=1}^{k}(k+1-i)\lceil\log i\rceil$$
$$= \sum_{i=1}^{k}(k+1)\lceil\log i\rceil - \sum_{i=1}^{k} i\lceil\log i\rceil$$
$$= (k+1)\sum_{i=1}^{k}\lceil\log i\rceil - \sum_{i=1}^{k} i\lceil\log i\rceil \tag{38}$$

Claim 2 $\sum_{i=1}^{k}\sum_{j=1}^{i-1}\lceil\log j\rceil = k\sum_{i=1}^{k}\lceil\log i\rceil - \sum_{i=1}^{k} i\lceil\log i\rceil$

Proof.

$$\sum_{i=1}^{k}\sum_{j=1}^{i-1}\lceil\log j\rceil = \sum_{i=1}^{k}\sum_{j=1}^{i-1}\lceil\log j\rceil - \sum_{i=1}^{k}\lceil\log i\rceil$$
$$= (k+1)\sum_{i=1}^{k}\lceil\log i\rceil - \sum_{i=1}^{k} i\lceil\log i\rceil - \sum_{i=1}^{k}\lceil\log i\rceil$$
$$= k\sum_{i=1}^{k}\lceil\log i\rceil - \sum_{i=1}^{k} i\lceil\log i\rceil \tag{39}$$

Claim 3 $\sum_{i=1}^{k}\sum_{j=1}^{k-i}\lceil\log j\rceil = \sum_{i=1}^{k}\sum_{j=1}^{i-1}\lceil\log j\rceil = k\sum_{i=1}^{k}\lceil\log i\rceil - \sum_{i=1}^{k} i\lceil\log i\rceil$

Claim 4 $\sum_{i=1}^{k} i\sum_{j=1}^{i}\lceil\log j\rceil = \sum_{i=1}^{k}\frac{(k-i+1)(k+i)}{2}\lceil\log i\rceil$

Claim 5 $\sum_{i=1}^{k} i\sum_{j=1}^{i-1}\lceil\log j\rceil = \sum_{i=1}^{k}\frac{(k+i+1)(k-i)}{2}\lceil\log i\rceil$

Claim 6 $\sum_{i=1}^{k} i\sum_{j=1}^{k-i}\lceil\log j\rceil = \sum_{i=1}^{k}\frac{(k-i+1)(k-i)}{2}\lceil\log i\rceil$

Using the summation results, we have:

$$\begin{aligned} E[\overline{C}] &= 1 + \frac{2}{k}\sum_{i=1}^{k}(\lceil\log i\rceil) - \frac{1}{k}\lceil\log k\rceil - \frac{2}{k^2}\sum_{i=1}^{k}(k-i)\lceil\log i\rceil \\ &= 1 - \frac{1}{k}\lceil\log k\rceil + \frac{2}{k^2}\sum_{i=1}^{k} i\lceil\log i\rceil \end{aligned} \tag{40}$$

Our final evaluation relies on the following theorems:

Theorem V.6 *Let (a_n) be an arbitrary sequence of variables. For $\forall n \in \mathbf{N}$, we have:*

$$\sum_{i=1}^{n}(a_i) = na_n - \sum_{i=i}^{k-1} i(a_{i+1} - a_i)$$

The proof is a simple mechanical verification. This theorem is useful whenever the difference $a_{i+1} - a_i$ $(1 \le i \le n-1)$ is easy to evaluate.

Theorem V.7 $\sum_{\alpha=1}^{k}(\lceil\log\alpha\rceil) = k\lceil\log k\rceil - 2^{\lceil\log k\rceil} + 1$

Proof. Replacing in Theorem V.6 (a_n) by $\lceil\log\alpha\rceil$ we have:

$$\sum_{i=1}^{k}\lceil\log i\rceil = k\lceil\log k\rceil - \sum_{i=1}^{k-1} i(\lceil\log(i+1)\rceil - \lceil\log i\rceil)$$

With $2^P \le n < 2^{P+1}$ for some $P \in \mathbf{N}$. We have $P \le \log n < P+1$ and consequently,

$$\lceil\log n\rceil = \begin{cases} P & n = 2^P \\ P+1 & 2^P < n < 2^{P+1} \end{cases}$$

The value of $\lceil\log(i+1)\rceil - \lceil\log i\rceil$ is either 0 or 1. The value is 1 whenever $i = 2^q$ for some $1 \le q < \lceil\log n\rceil$, and 0 everywhere else.

Thus,

$$\sum_{i=1}^{k-1} i(\lceil\log(i+1)\rceil - \lceil\log i\rceil) = \sum_{q=0}^{\lceil\log k\rceil-1} 2^q = 2^{\lceil\log k\rceil} - 1$$

The conclusion follows.

Corollary V.8 $\sum_{\alpha=1}^{k}(\lfloor\log\alpha\rfloor) = k\lfloor\log k\rfloor - 2^{\lfloor\log k\rfloor+1} + 2 + \lfloor\log k\rfloor$

Proof. Note that the value of $\lfloor\log(i+1)\rfloor - \lfloor\log i\rfloor$ is either 0 or 1. The value is 1 whenever $i = 2^q - 1$ for some $1 \leq q < \lfloor\log n\rfloor$, and 0 everywhere else.

Thus,

$$\sum_{i=1}^{k-1} i(\lfloor\log(i+1)\rfloor - \lfloor\log i\rfloor) = \sum_{q=1}^{\lfloor\log k\rfloor} 2^q - 1 = 2^{\lfloor\log k\rfloor+1} - 2 - \lfloor\log k\rfloor$$

The conclusion follows.

Theorem V.9 $\sum_{i=1}^{k-1} i^2(\lceil\log(i+1)\rceil - \lceil\log i\rceil) = \frac{4^{\lceil\log k\rceil-1}}{3}$ *With* $2^\alpha \leq k < 2^{\alpha+1}$ *for some natural number* α.

Case 1: $k = 2^\alpha$

Notice that the difference $\lceil\log(i+1)\rceil - \lceil\log i\rceil$ Is non-zero only if $i = 2^\beta$ for some $\beta \leq \alpha - 1$. The value of the corresponding item is $2^{2\beta} = 4^\beta$. Thus, the value of the sum is:

$$\sum_{\beta=0}^{\alpha-1} 4^\beta = \frac{4^{\beta+1}-1}{3} = \frac{4^\alpha - 1}{3} = \frac{4^{\lceil\log k\rceil}-1}{3}$$

Case 2: $2^\alpha < k < 2^{\alpha+1}$ (we have $\lceil\log k\rceil = \alpha + 1$)

As before, the term is 4^β. Thus the sum becomes:

$$\sum_{\beta=0}^{\alpha} 4^\beta = \frac{4^{\beta+1}-1}{3} = \frac{4^{\alpha+1}-1}{3} = \frac{4^{\lceil\log k\rceil}-1}{3}$$

As claimed.

Corollary V.10 $\sum_{i=1}^{k-1} i^2(\lfloor\log(i+1)\rfloor - \lfloor\log i\rfloor) = \frac{4^{\lfloor\log k\rfloor+1}-1}{3} - 2^{\lfloor\log k\rfloor+2} + \lfloor\log k\rfloor + 3$

Theorem V.11 $\sum_{i=1}^{k} i\lceil\log i\rceil = \frac{1}{2}\left[k^2\lceil\log k\rceil + k\lceil\log k\rceil - 2^{\lceil\log k\rceil} + 1 - \frac{4^{\lceil\log k\rceil}-1}{3}\right]$

Proof. Using Theorem V.6, we have:

$$\begin{aligned}\sum_{i=1}^{k} i\lceil\log i\rceil &= k^2\lceil\log k\rceil - \sum_{i=1}^{k-1} i((i+1)\lceil\log(i+1)\rceil - i\lceil\log i\rceil) \\ &= k^2\lceil\log k\rceil - \sum_{i=1}^{k-1} i^2(\lceil\log(i+1)\rceil - \lceil\log i\rceil) - \sum_{i=1}^{k-1} i\lceil\log(i+1)\rceil \qquad (41)\end{aligned}$$

Consequently

$$\sum_{i=1}^{k} i\lceil\log i\rceil + \sum_{i=1}^{k-1} i\lceil\log(i+1)\rceil = k^2\lceil\log k\rceil - \sum_{i=1}^{k-1} i^2(\lceil\log(i+1)\rceil - \lceil\log i\rceil)$$

Eventually

$$\sum_{i=1}^{k} i\lceil\log i\rceil + \sum_{i=1}^{k-1}(i+1)\lceil\log(i+1)\rceil - \sum_{i=1}^{k-1}\lceil\log(i+1)\rceil$$
$$= k^2\lceil\log k\rceil - \sum_{i=1}^{k-1} i^2(\lceil\log(i+1)\rceil - \lceil\log i\rceil)$$

Note that:

$$\sum_{i=1}^{k-1}(i+1)\lceil\log(i+1)\rceil = \sum_{i=2}^{k} i\lceil\log i\rceil = \sum_{i=1}^{k} i\lceil\log i\rceil$$

And

$$\sum_{i=1}^{k-1}\lceil\log(i+1)\rceil = \sum_{i=2}^{k}\lceil\log i\rceil = \sum_{i=1}^{k}\lceil\log i\rceil$$

We have

$$2\sum_{i=1}^{k} i\lceil\log i\rceil = \sum_{i=1}^{k}\lceil\log i\rceil + k^2\lceil\log k\rceil - \sum_{i=1}^{k-1} i^2(\lceil\log(i+1)\rceil - \lceil\log i\rceil)$$

Applying Theorem V.9 we have

$$\begin{aligned}\sum_{i=1}^{k} i\lceil\log i\rceil &= \frac{1}{2}\left[k^2\lceil\log k\rceil + \sum_{i=1}^{k}\lceil\log i\rceil - \frac{4^{\lceil\log k\rceil}-1}{3}\right] \\ &= \frac{1}{2}\left[k^2\lceil\log k\rceil + k\lceil\log k\rceil - 2^{\lceil\log k\rceil} + 1 - \frac{4^{\lceil\log k\rceil}-1}{3}\right] \qquad (42)\end{aligned}$$

Corollary V.12 $\sum_{i=1}^{k} i\lfloor\log i\rfloor = \frac{1}{2}\left[k^2\lfloor\log k\rfloor + k\lfloor\log k\rfloor + 2^{\lfloor\log k\rfloor+1} - 1 - \frac{4^{\lfloor\log k\rfloor+1}-1}{3}\right]$

Now, we can evaluate the value of $E[\overline{C}]$:

$$\begin{aligned}E[\overline{C}] &= 1 - \frac{1}{k}\lceil\log k\rceil + \frac{1}{k^2}\left[k^2\lceil\log k\rceil + k\lceil\log k\rceil - 2^{\lceil\log k\rceil} + 1 - \frac{4^{\lceil\log k\rceil}-1}{3}\right] \\ &= 1 + \lceil\log k\rceil - \frac{3*2^{\lceil\log k\rceil} + 4^{\lceil\log k\rceil} - 4}{3k^2} \qquad (43)\end{aligned}$$

Theorem V.13 $E[\overline{C}] = 1 + \lceil\log k\rceil - \frac{4^{\lceil\log k\rceil}+3*2^{\lceil\log k\rceil}-4}{3k^2}$

Similarly, we can evaluate $E[\underline{C}]$ using the following theorem:

Theorem V.14 $E[\underline{C}] = 1 + \lfloor\log k\rfloor - \frac{4^{\lfloor\log k\rfloor+1}-3*2^{\lfloor\log k\rfloor+1}+2}{3k^2}$

In the worst case scenario, the searching awake slot number is $k(\lceil\log k\rceil - 1)$ plus 1. The overall time slot T needed for training is:

$$T = L + k(\lceil\log k\rceil - 1) + 1 \qquad (44)$$

V.5.3 Asynchronous Training

In section V.3 ([113]), we propose a fully asynchronous training protocol for massively-deployed sensor networks. In the protocol, we assume the value of k is known by the sensors. Similar to analysis for the linear search scheme, we can build a searching awake time matrix in Table 7

Table 7: Asynchronous searching awake time matrix

$i\backslash s$	1	2	3	$\cdots$	$k-2$	$k-1$	k
1	0	1	2	$\cdots$	$k-3$	$k-2$	$k-1$
2	$k-1$	1	2	$\cdots$	$k-3$	$k-2$	$k-1$
3	$k-1$	$k-1$	1	$\cdots$	$k-4$	$k-3$	$k-2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-2$	4	5	6	$\cdots$	1	2	3
$k-1$	3	4	5	$\cdots$	$k-1$	1	2
k	2	3	4	$\cdots$	$k-1$	$k-1$	1

We can evaluate the expected awake slot number using the following equation:

$$E[C] = \frac{\sum_{i=1}^{k-1}(i) - 1 + k - 1}{k} + 1 \tag{45}$$

By simplifying equation (45), we have

$$E[C] = \frac{k}{2} + \frac{3}{2} - \frac{2}{k} \tag{46}$$

In the worse case scenario, the total time slots needed for asynchronous training is $\lceil k/d \rceil * L$

V.5.4 Summary

We summaries the protocol performance in term of total time slot and average awake time slot in Table 8

Table 8: Performance summary

	Total slots	Awake slots
SYNC training	$L + k - 1$	$\lceil \log k \rceil + 1$
ASYNC training	$\lceil \frac{k}{d} \rceil * L$	$\frac{k}{2} + \frac{3}{2} - \frac{2}{k}$
Linear position search	$L + k - 1$	$\frac{k}{4} + 2\frac{1}{6} - \frac{5}{4k} - \frac{7}{6k^2} + \frac{1}{k^3}$
Binary position search	$L + k(\log k - 1) + 1$	$\leq \lceil \log k \rceil + 1 - \frac{3*2^{\lceil \log k \rceil} + 4^{\lceil \log k \rceil} - 4}{3k^2}$

V.6 SIMULATION RESULTS

In our simulation, sensors are deployed uniformly at random in a field of size 640m x 640m. A sink is placed in the middle of the field at (320, 320). The number k of coronas is 32 and each corona has a width of 10 meters. Hence, the length of a $k-$ *cycle* is 32 time slots (a slot is 10 milliseconds). The sensors wake up at random to simulate the asynchronous effect. For each sensor, the first wakeup time is generated uniformly at random in the interval $(0, T_0)$, where $T_0 = 32$ for the asynchronous training and $T_0 = 100$ for the hybrid training (in fact, for the purpose of the simulation, T_0 is taken to be 1,000,000 and 320,000 because the time unit is taken to be the microsecond, so T_0 is equal to 100 and 32 time slots, respectively).

V.6.1 Asynchronous Training

By equation (14) in Section V.3.2, if we want all sensors to be trained in 3 s-cycles, i.e, $n = 3$, then d must equal 11. If we choose $m = 2$, then by Theorem V.1, the length L of the s-cycle will be $L = mk + d = 75$ and the total time to train all the sensors will be $2L + k + d = 150 + 32 + 11 = 193$ time slots. Our simulation confirms our theoretical prediction as showed in Fig. 40. In Fig. 40, one can see that the sensors are evenly trained in the 3 s-cycles. More specifically, in the first training interval, approximately 33.3% of the sensors is trained. The same percentage of the sensors is trained in the second and the third training intervals. In the figure, the first training interval is from 11 to 43 because each sensor wakes up randomly at a time between 0 and 32 time slot and listens for 11 time slots. In the first training interval, the earliest time that a sensor is trained is at time 11 and the latest time is 43. In the second training interval, the earliest time that a sensor is trained is at

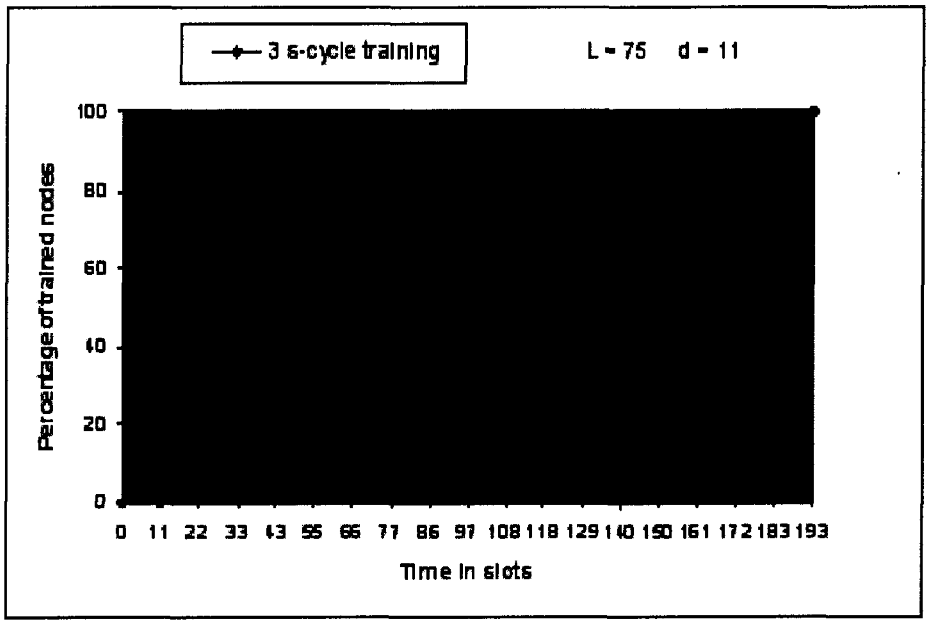


Fig. 40: Illustrating 3 s-cycle training.

time 11+75 = 86, the latest time is 43+75 = 118, and so on.

Fig. 41 and 42 show the total training time (in slots) with the different values of d (the value of L remain fixed at 75). It is clear that 11 is the optimal value for training the sensors. On the one hand, if the value of d is smaller than 11, then it takes much longer to train the sensors. The overall awake time is larger than 33 (33 is the overall awake time for 3 s-cycles when $d = 11$). For instance, in Fig. 41 we can see that if $d = 5$, the total time slots needed to train the sensors is around 1500 (20 s-cycles), which requires overall 100 (20*5) awake time slots to train the sensors. On the other hand, if the value of d is larger than 11 (see Fig. 42), the overall awake time is still larger than 33 because a larger awake interval does not help much. For instance, when $d = 21$, the sensors need 2 s-cycles to get trained, which makes the overall awake time equal to 42.

Fig. 43 shows that the length of s-cycle is also very important for sensor training. The larger is the greatest common divisor of L and k, the worse for training the sensors. For example, if $L = 104$ and $k = 32$, the greatest common divisor of L and k is 8, which means the sensors has to be trained in 4 s-cycles. Any sensor is not

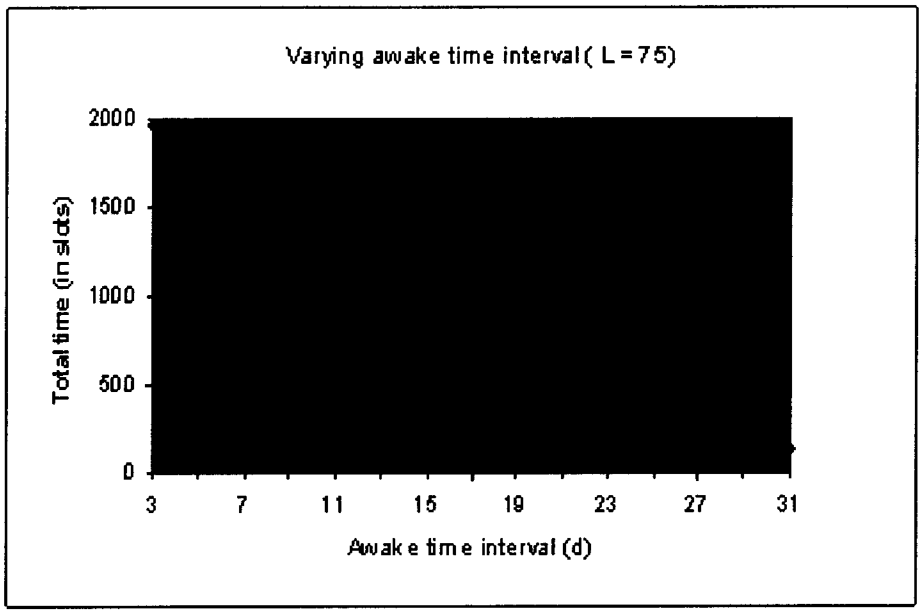


Fig. 41: Varying awake time interval – the overall picture.

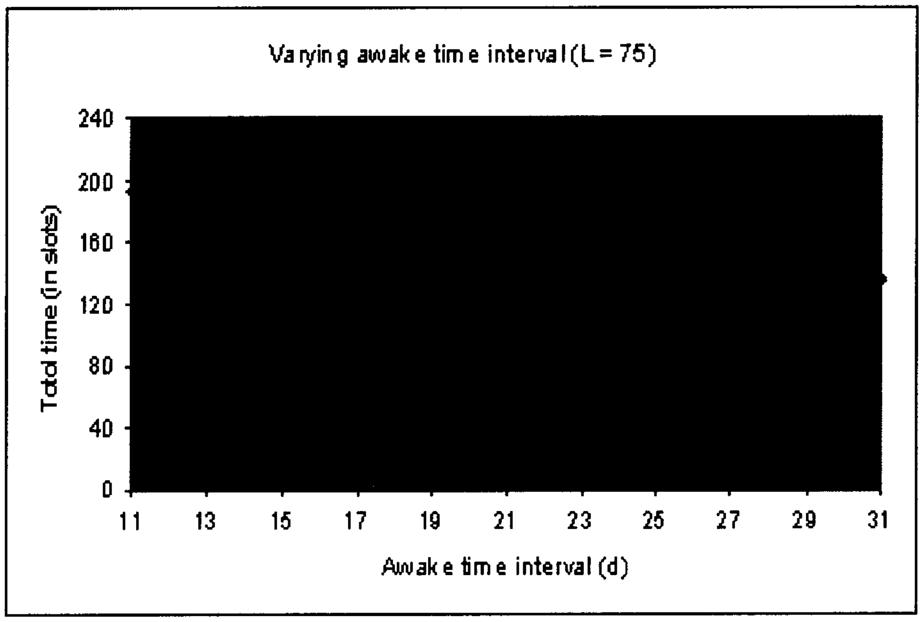


Fig. 42: Varying awake time interval – the details.

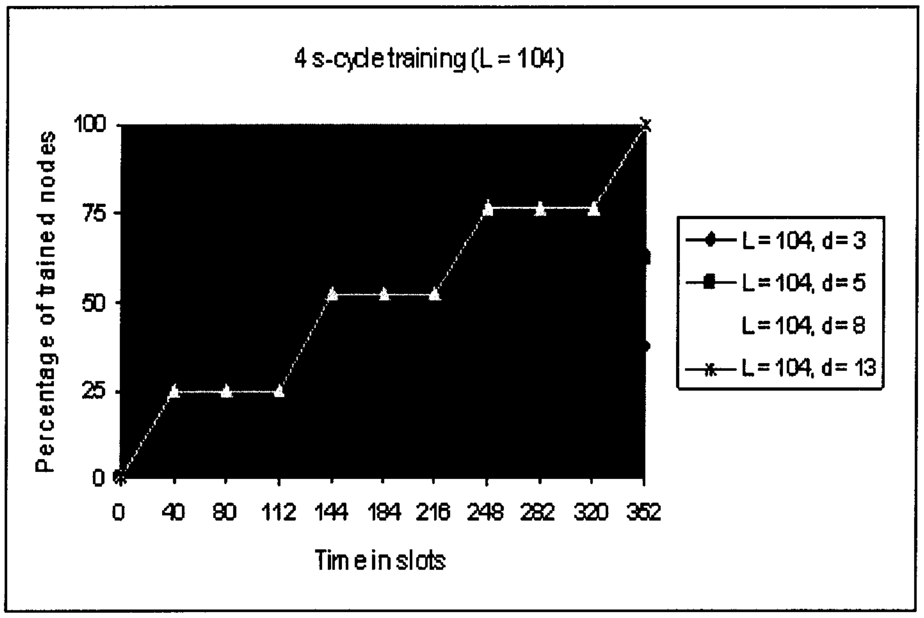


Fig. 43: The importance of the s-cycle length.

trained in 4 s-cycles will never be trained. It is illustrated in the Fig. 43 that only 37.5% and 62.5% sensors are trained when $d = 3$ and $d = 5$, respectively. This makes sense because only 37.5% and 62.5% of 32 is covered in 4 s-cycles when $d = 3$ and $d = 5$, respectively. In this case, d has to be at least 8 to cover 32 in 4 s-cycles and all sensors will be trained.

Our simulation results show that if k is known, we can decide the number n of s-cycles needed to train the sensors and we can compute the optimal values of d and L s a function of both n and k, using Theorem V.1. However, if k is not known in advance, we are not able to compute the optimal value of d and L. One important conclusion that the simulation reveals is that the value of L should be a prime number in order to minimize the greatest common divisor of L and k.

V.6.2 Hybrid Training

Fig. 44, 45 and 46 show the performance of the linear search scheme and the binary search scheme. In both schemes, the value of k is set to 32. The simulation program was executed 100 runs for different simulation settings and the averages of both total time slots T and awake time slots T_a are computed. Fig. 44 shows that the values of T from different runs are very stable and the sensors are evenly trained during the total training time T period. On the other hand, the awake time slots (Fig. 45 and 46 varied when the number of sensors is in a small range (100 – 500). The value of T_a becomes stable and reaches the analytic anticipation when the number of sensors is larger than 1500.

As anticipated, the total time slot number needed for training in the linear search scheme is $T = L + k - 1 = 100 + 32 - 1 = 131$. And the expected awake time slot number is close to $E(C) = \frac{k}{4} + 2\frac{1}{6} - \frac{5}{4k} - \frac{7}{6k^2} + \frac{1}{k^3} = 10.126$ when the number of sensors is large enough. The total time slot number needed for training in the binary search scheme is $T = L + k(\log k - 1) + 1 = 229$. And the expected awake time slot number is close to $5.3 \leq E[\overline{C}] = 5.637$ when the number of sensors is large enough.

In Fig. 47 we plotted the values of $E[\underline{C}]$, $E[C]$ and $E[\overline{C}]$ for various k from $k = 32$ to $k = 63$. The simulation result shows that the difference between the $E[\overline{C}]$ and $E[C]$ is less than 0.5.

V.7 SUMMARY

Endowing sensors with coarse-grain location awareness, a task referred to as *training* is essential in numerous applications. The main contribution of this work was to propose asynchronous training protocols for sensor networks. We showed analytically that in spite of the lack of synchronization, individual sensors are trained energy-efficiently. The analytical results have been confirmed by extensive simulation.

There is a problem that still remains open: the synchronous training protocol have individual sensor be awake for only $\log k$ time but consumes a lot of energy in the toggling between sleep and wake periods. On the other hand, the asynchronous training protocol can train the sensors as efficient as the synchronous training protocol. However, the asynchronous protocol forces the sensors to be trained for longer periods to avoid frequent transitions from sleep to wake periods. Striking the right balance between the two promises to be an exciting area of further work.

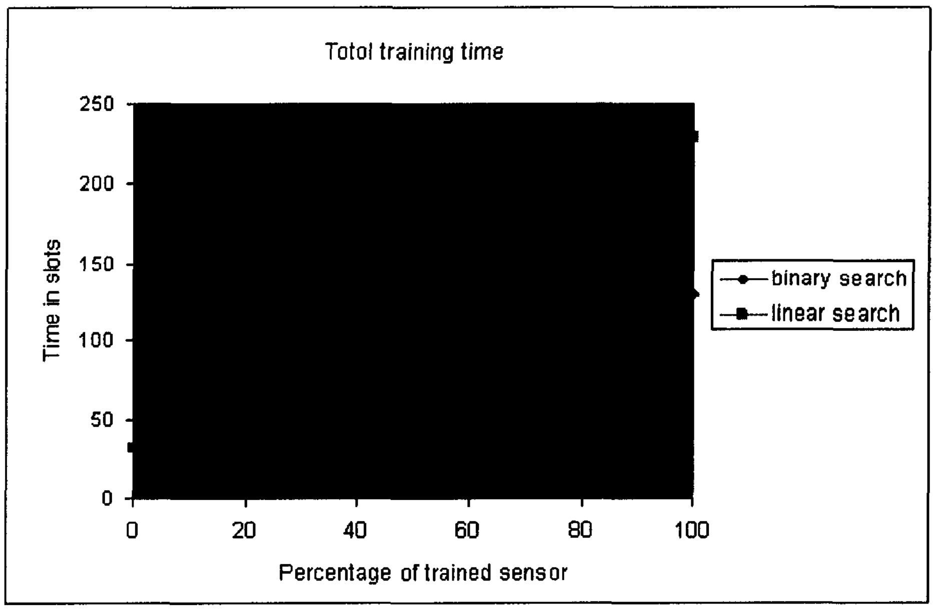


Fig. 44: Total training time for linear search.

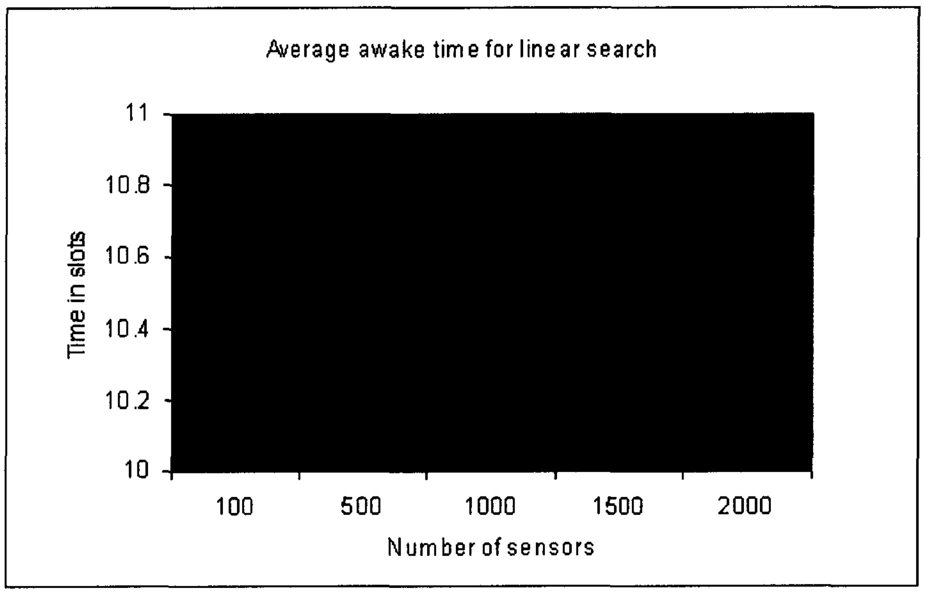


Fig. 45: Average awake time for linear and binary search.

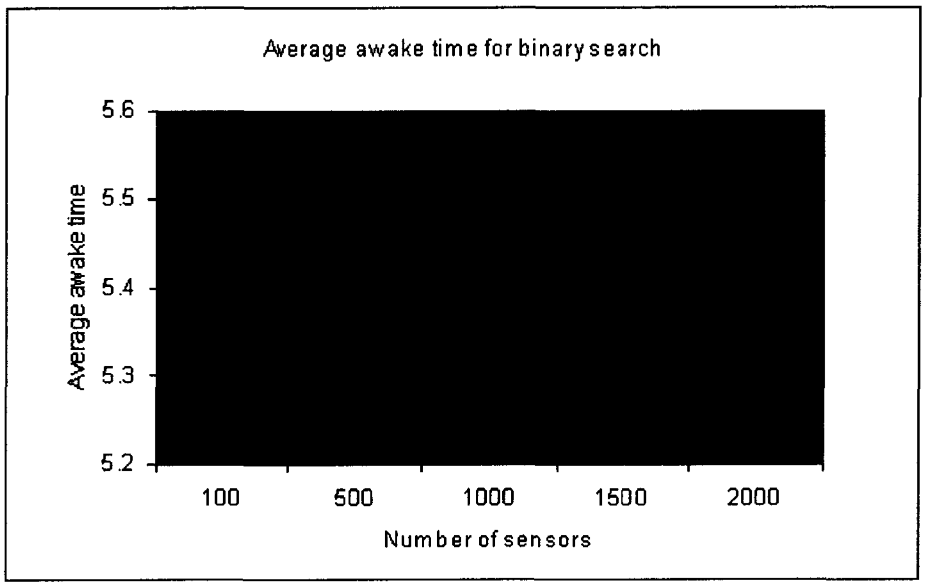


Fig. 46: Average awake time for binary search.

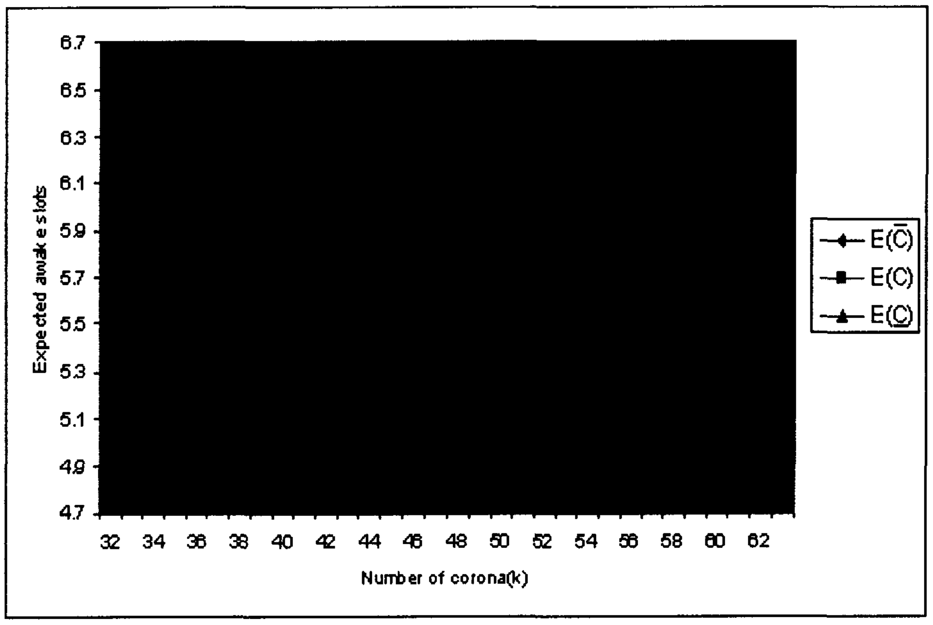


Fig. 47: Awake time boundary for binary search.

CHAPTER VI
USING THE INFRASTRUCTURE: ROUTING

VI.1 INTRODUCTION

A routing protocol for the sensor network is desired to be in its simplest form, meaning that it should present minimum control message overhead. Specifically, a routing protocol should utilize minimum energy, bandwidth and memory to form and maintain routes from sensors to the sinks. Normally a shortest path spanning tree rooted at a sink is constructed for routing. The shortest path spanning tree is easy to construct but difficult to maintain if the protocol only keeps the smallest route information and utilize minimum resources to update topological changes. When the network topology is changed, control messages are transmitted in the network. In order to maintain a shortest path tree structure, not all topology changes are needed to be advertised. For instance, if a non tree link is broken, the information can be simply ignored by the protocol. On the other hand, if a tree link is broken or a new link that will change some node's tree level, is formed, control messages have to be sent to allow the protocol to reconstruct the shortest path tree. The key of the routing protocol efficiency relies on the control message overhead.

The sinks are in charge of tasking the sensors and collecting sensing information from the sensors. The former is normally accomplished by a one-to-many operation (broadcasting or multicasting) starting at the sink and the latter is accomplished by a many-to-one operation starting at the individual sensors.

In Chapter IV, we presented a novel self-organization protocol, which establishes a two-level hierarchical infrastructure, as showed in Fig. 48. The infrastructure can significantly benefit the one-to-many and many-to-one operations as well.

At the top level of the hierarchy, clusters form a virtual network. The virtual network consists of cluster leaders (for virtual nodes) and active gateways (for virtual links). Since the forwarding of data and control messages only involves the virtual network, significant communication overhead is eliminated. In this chapter, we present a novel resource-efficient routing protocol for the sensor network. The protocol maintains a shortest path spanning tree on top of the virtual network without the knowledge of network topology or whole path to the destination. The protocol

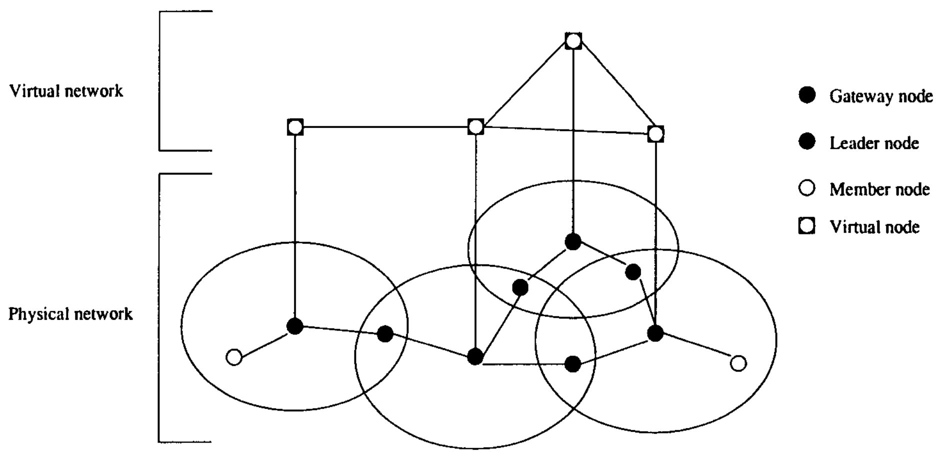


Fig. 48: Two level network hierarchy.

only stores local information to reconstruct the tree when a tree link fails. Moreover, the protocol presents minimum control message overhead. In this chapter, we use the term node and virtual node interchangeably to represent the clusters in the infrastructure.

The chapter is organized as follows: Section VI.2 gives the background knowledge and related work. Section VI.3 describes our new routing protocol in detail. Section VI.4 presents the routing algorithm. Section VI.5 gives the correctness proof. Section VI.6 presents examples of the operation of the new algorithm. And Section VI.7 provides the performance comparison between several well known routing algorithms and the new algorithm of Section VI.4.

VI.2 BACKGROUND

VI.2.1 Problem Statement

Most existing routing protocols usually do not fulfill the following requirements of the sensor networks.

- *Supporting the correlated and periodic traffic:*

Reactive routing techniques, which include most routing protocols ([34, 53, 68, 76, 79, 80, 106] developed for mobile ad hoc networks, are deemed inappropriate in sensor networks.

- *Utilizing very limited network resources:*

  Localized routing protocols (e.g. Distance vector protocol [35]) are better candidates than global routing protocols (e.g. Link state protocol [51]).

- *Supporting efficient network flooding or limited flooding mechanism:*

  Constructing a communication infrastructure (e.g. cluster, backbone) reduces the flooding deficiency listed in [2].

- *Demanding routing techniques without global unique identifiers:*

  Global topology information is not only expensive but also impossible to get. For instance, since individual sensors do not have unique IDs, the route information stored in each sensor can not specify the entire path from the source to the destination ([59]).

Due to these requirements, special multi-hop routing protocols between the sensors and the sink are needed. The routing protocols should employ localized algorithms to construct and maintain a shortest hop spanning tree rooted at a sink. The difficulty is to reconstruct the tree when the network topology changes, while minimizing the control message overhead.

One of localized routing protocols is the famous distance vector routing protocol (DVP [35, 104]). In the DVP, in order to reconstruct the tree after topology changes, each node also keeps the distances from its neighbors to the root. These information are used to compute its tree level when the topology changes.

The expense of utilizing only local information in the DVP is that it suffers from the route loop and slow convergence problems. Basically, the route loop and slow convergence are the different faces of the same problem. In DVP, a node only keeps neighbor route information and it has not knowledge of the network topology or the whole path to a destination. Each node computes its shortest path to a destination based on limited route information received from its neighbors and the route information could be stale information. For example, in Fig. 49, assume link (A, B) fails. After the link failure, node B chooses node C as its next hop to

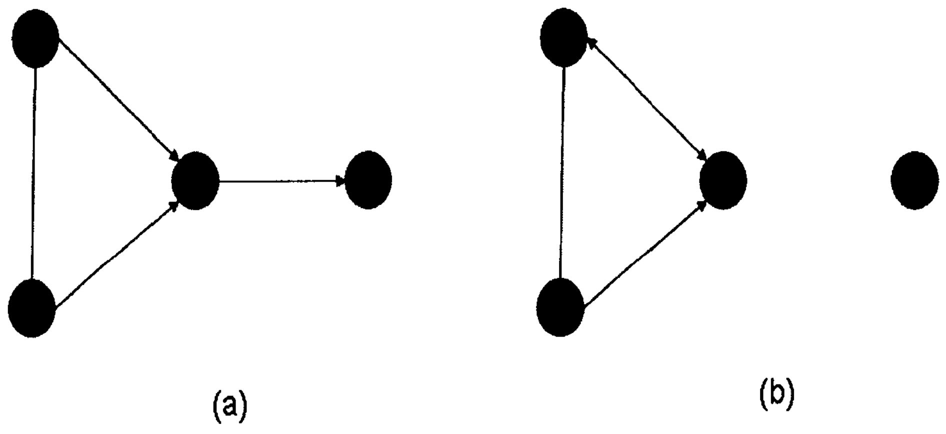


Fig. 49: Demonstrating the route loop when link(a, b) fails.

destination A, which cause a loop (B, C, B) and also causes count to infinity problem because the wrong route information are advertised again and again. (Bad news is advertised slowly in the DVP).

The pioneering work for solving the route loop and slow convergence problems is dated back to [65] and [52]. In [65], a tree link broken event is advertised in two directions: up-tree advertisement is used to clean up the invalid routes caused by the broken link. Down-tree advertisement is used to inform the sink (the root). After receiving the advertisement, the sink will initialize a "new update cycle", which will repair the shortest path tree and guarantee loop-free route table at all times. The "failsafe" feature of the algorithm is at the expense of large control message overhead since in each "new cycle" the control messages are propagated throughout the whole network.

The Jaffe-Moss algorithm in [52] is an instances of diffusing computation which is invented by E.W. Dijkstra and C.S. Scholten in [15]. The diffusing computation started by a node grows be sending queries and shrinks by receiving replies along the sub-tree rooted at the source of the computation. Using the diffusing computation, the tree recovery is sustained at "local": the nodes affected by the failure. The idea is to inform the affected nodes the topology change before they compute their new shortest path. The affected nodes are "frozen" and wait for the "unfrozen" message from all its neighbors. In the protocol, the control messages are not needed to be advertised outside the sub-tree rooted at the initial node (where the tree link fails). However, control messages are sent back and forth from the initial node to sub-tree leaves several times before the route computing starts. In order to reduce the control message overhead, it is preferred that the control messages are only sent once to clean up the invalid routes affected by the failure. And each node makes decisions on its own instead of relying on decisions from some other node.

VI.2.2 Related Work

The power consumption of the communication is significantly greater than that of the computations in the sensor networks. As a consequence, the data-centric routing technique [37, 41, 49] is introduced to replace the communication with the computation. Power aware routing techniques are proposed in [10, 11, 89, 117] to steer traffic away from low energy sensors. Other efforts include designing multi-path routing protocols [26] to overcome the sensor failure issue and the random routing protocols [7, 56] to fight the flooding deficiencies.

The Directed diffusion data dissemination paradigm is proposed in [41] where the sinks query for information by disseminating an interest. An interest is simply a range of values for one or more attributes. Each sensor stores the interest entry in its cache. As the interest is propagated throughout the sensor network, the gradients from the sensors to the sinks are set up. Data flow back along the interest's gradient path. Also, the sinks must refresh and reinforce the interest when they start to receive data.

Directed diffusion [41] requires periodic low-rate flooding of data in order to allow recovery from failure. To route around failed sensors, multi-path routing protocols are proposed in [26]. Of the many possible designs of multi-path routing, two schemes are considered in [26]: Disjoint multi-path and braided multi-path. Disjoint multi-path scheme create number of routes with no common branches from a sensor to a sink. The braided multi-path scheme relaxes the requirement for disjointed-ness. The authors compared the performance of the two schemes based on resilience (a measure of the likelihood that, when the shortest path fails, an alternate path is available between a sensor and a sink) and maintenance overhead (measure of the energy required to maintain alternative paths). Simulation results show that braided multi-path expends only 33 percent of the energy of disjoint path for alternate path maintenance, and has 50 percent higher resilience to isolated failures.

Rumor routing [7] attempts to alleviate flooding deficiencies in directed diffusing, where flood queries across the entire network. The idea is to create paths leading to each event (An event is an abstraction, identifying anything from a set of sensor readings, to the node's processing capabilities); whereas event flooding creates a network-wide gradient field. In this way, when a query is generated that it can be sent on a random walk until it finds the event path, instead of flooding it across the network.

The SAR algorithm [98] aim to solve energy unbalance problem in the sensor network, which is: the bulk of traffic in sensor network is the data sent from sensors to the sink. This will put significant strain on the energy resources of the sensors near the sink, making that neighborhood more susceptible to energy depletion and failure.

The SAR algorithm creates multiple trees where the root of each tree is a one hop neighbor from a sink. Most sensors in the network belong to multiple trees. This allows a sensor to choose a tree to relay its information back to the sink. The SAR algorithm selects the route based on maximum available power matrix and additive QoS metric, and the packet's priority level.

A family of adaptive protocols (SPIN [37]) is designed to address the deficiencies of flooding by negotiation and resource-adaptation. Sensors name their data using high-level data descriptors, called meta-data. Sensors use meta-data negotiations to eliminate the transmission of redundant data throughout the network. In addition, sensors can base their communication decisions both upon application-specific knowledge of the data and upon knowledge of the resources that are available to them. This allows the sensors to efficiently distribute data given a limited energy supply. SPIN has three types of messages, i.e., ADV, REQ, and DATA. A sensor broadcasts an ADV message containing data descriptor. If a neighbor sensor is interested in the data, it needs to send a REQ message for requiring a DATA message.

VI.3 NEW ROUTING PROTOCOL

Our goal is to design a routing protocol that keeps minimum routing information, utilizes simplest control message form and creates smallest amount of control message overhead. In other word, the protocol uses minimum network resources to construct and maintain a simple shortest hop spanning tree rooted at a sink. At each node, the minimum necessary route information needed is the parent pointer (p) and the tree level (d). The parent pointer indicates the node's parent in the tree and the tree level indicates the distance (in term of hops) to the root (the sink).

The basic idea is simple: the routing protocol should not compute the shortest path using the stale information, which causes unnecessary control message overhead. Hence, the invalid routes caused by broken links should be cleaned up first.

We first define notations used in our protocol before describing the detail operations of the protocol. The sensor network is modeled as a directed graph $G = (N, E)$, where N is a set of nodes and E is a set of edges, which are the directed links between nodes. For a given sink s, each node maintains a route entry $(d[v], p[v], b[v])$, where $d[v]$ is the tree level of v, $p[v]$ is the parent pointer of v, and $b[v]$ is the broken tree level of v. A path from node d to s is a sequence of nodes d, e, $\ldots$, m, n, $\ldots$, t, s in which (d, e), $\ldots$, (m, n), $\ldots$, (r, s) are edges in the path. The tree level (or the distance to the sink) of node v is measured as the number of the edge of a path between node v and sink s. If there is no path between node v and sink s, the tree level $d[v]$ is ∞. If there is no link failure, the broken tree level b[v] is ∞, too.

At any given time, the path formed by consecutive parent pointer defines routes to the sink. Node i is said to be up-tree node of node k if the directed path P is from node i to node s includes node k. Node k is said to be down-tree node of node i. The proposed protocol reconstructs the shortest path tree after link failures and/or additions of links to the network. Note that a node failure can be represented as the failure of all its links, and similarly, an addition of node can be represented as additions of links. Therefore, we do not pay special attention to node failures and node additions [65].

In the protocol, a node informs its neighbors about its route information by sending two types of messages: Positive Update (*PU*) message and Negative Update (*NU*) message. A *PU* message has the form of $(d[v], p[v])$ and it is initiated by link additions and route recoveries. The sink engagement to the network can be viewed as several link additions to the graph G. It will trigger the *PU* message generation and the protocol execution. Each node in the network initializes its route entry as (∞, nil, ∞). We assume the sink s will engage with the network by sending a *PU* $(0, nil)$ message to its neighbors. After receiving the messages, the neighbors of the sink will update their route entry to $(1, s, \infty)$ and send their own *PU* $(1, s)$ messages.

When a node receives a *PU* message, it reevaluates its tree level to the sink and updates its tree level and parent pointer if its tree level is lower than its previous tree level, and sends a *PU* message to all its neighbors. The *PU* messages are propagated throughout the network until the spanning tree is formed. When a link (v, u) is added in the network, node v and u will send *PU* messages if either of the nodes has a route to the sink. Otherwise, no action will be taken.

A *NU* message has the form of $(b[v], p[v])$ and it is initiated by a link failure. Each node in the spinning tree keeps track its parent node. If a child node does not receive some confirm message from its parent with a specified time period, called the maximum link delay (*MLD*), the link is considered to be broken. Hence, any control message is guaranteed to be received within a *MLD* or a link failure is detected. Normally a link failure is detected by a link-level protocol. Our routing protocol only deals with the tree link failure. Other link failure can be simply ignored. When a node v detects the link to its parent $p[v]$ fails, it sends a *NU* message $(b, p[v])$ to all its neighbors, where $b = d[v]$ and updates its tree level $d[v] = \infty$ and its parent pointer to *nil*. The children of node v will do the same and eventually all the invalid routes caused by the failure will be cleaned up. The value of broken tree level $b[v]$ remains the same during the *NU* message propagation.

The route entry $(d[v], p[v], b[v])$ can have four forms that represent four states of the protocol. As already mentioned, the protocol assumes each route entry is initialized as (∞, nil, ∞) for no-route state (NS) to indicate the node has no route knowledge to the sink s. Other three forms of route entries for a node v are $(d[v], p[v], \infty)$ where $0 \leq d[v] < \infty$ for formative state (FS), $(\infty, nil, b[v])$ where $0 < b[v] < \infty$ for broken state (BR) and $(d[v], p[v], b[v])$ where $\infty > d[v] > b[v] > 0$ for wait state (WS). The formative state indicates that there is a route through $p[v]$ to the sink s and its tree level is $d[v]$ ($d[v] = 0$ only if $v = s$). The broken state indicates the route to the sink s is broken at tree level $b[v]$. The wait state represents a transient state. It indicates that node v has a route via $p[v]$ and its tree level is $d[v]$. Meantime, it also shows that node v is informed that the original shortest path tree is broken at tree level $b[v]$. If $d[v] \leq b[v]$, node v can be certain that its tree level $d[v]$ is not affected by the broken tree link. So it is safe for node v to send a *PU* message to its neighbors. On the other hand, if $d[v] > b[v]$, node v is not able to decide whether its tree level is valid or not, which implies that node v should not send a *PU* message to its neighbors. Fig. 50 gives an example of the wait route state. Fig. 50 (a) shows the original spanning tree. FiG 50 (b) shows that nodes F and H are in wait states after the tree link (A, C) fails. We assume the *NU* message from node C via node G to node H is received earlier than the message via node D. When node H receives the message, it enters wait state since $d[H] > b[H]$. Node F does the same although its route is valid for the moment when it receives the *NU* message from node C.

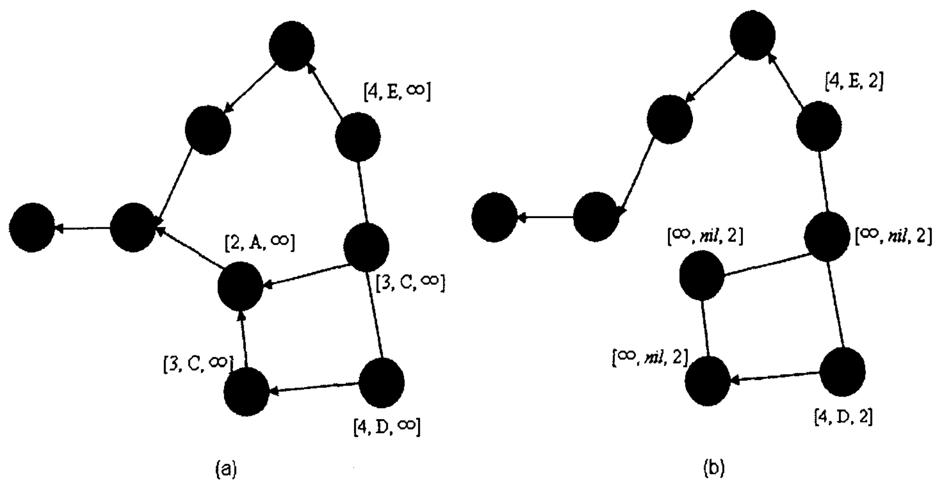


Fig. 50: Demonstrating wait route entry.

In the protocol, when $b[v] < \infty$, $b[v]$ is increased by 1 for every *MLD*. A safe reply condition $SRC(= d[v] - b[v]) \leq 0$ is used to check if a node can leave its wait state. In the Fig. 50 (b), node H will eventually receive a *NU* message from node *D*, which will update $d[H]$ to ∞. It makes node *H* go to the broken state. On the contrary, node *F* will wait until $b[F] = 4$, which makes $SRC(= d[F] - b[F]) \leq 0$. It makes node *F* go to the formative state and broadcast a *PU* message. In the example, we see that a node can leave the wait state in two cases. It may enter the broken state when it receives a *NU* message from its parent p or the link to its parent fails. Or it enter the formative state after a time period of $(d - b) * MLD$.

In [52], it is proved that only the tree level to the sink increases may cause a route loop. In our protocol, the only case that a tree level will increase is when a tree link fails and causes all its up-tree nodes updating their tree level to ∞ by sending *NU* messages. No route loop will form if a node does not send a *PU* message when

Table 9: Variables of the routing algorithm

Variable name	Domain of value	Meaning
$d[v]$	$0 \leq d[v] \leq \infty$	The tree level of node v
$p[v]$	Any other node or *nil*	The parent pointer of node v
$b[v]$	$0 \leq p[v] \leq \infty$	The broken tree level received

$SRC > 0$.

VI.4 THE ALGORITHM

In this section, we describe the exact algorithm performed by an arbitrary node v as its part of the protocol. Table 9 lists all the variables used by the algorithm at node v and the domain of their values. Table 10 lists the states in which node v can be. The algorithm processes five events, four of which may cause a node state change, are: an *PU* message received, an *NU* message received, a time out and a link failure detected. The link addition event may cause a node to send a *PU* message but will not cause the node to change its state.

Note that, for simplicity, we assume a control message is broadcast and will be received by all the neighbors of the sender.

Algorithm 1 *A). When receiving a PU message ($d[u], p[u]$) from node u and $p[u] \neq v$, node v does the following: 1). Sets $d = d[v]$. If $d[v] > d[u]+1$, sets $d[v] = d[u]+1$, $p[v] = u$. 2). If $d[v] < d$ and $b[v] = \infty$, broadcasts a PU message ($d[v], p[v]$). 3). If $b[v] < \infty$ and $d[v] \leq b[v]$ ($SRC \leq 0$), broadcasts a PU message ($d[v], p[v]$) and sets $b[v]to\infty$.*

B). When receiving a NU message ($b[u], p[u]$) from leader u and $p[u] \neq v$, node v does the following: 1). Sets $d = d[v]$, $p = p[v]$. If $p[v] == u$, sets $d[v] = \infty$, $p[v] = nil$. If $b[u] < b[v]$, sets $b[v] = b[u]$. 2). If $d < \infty$ and $d[v] = \infty$, broadcasts a NU message ($b[v], u$). 3). If $d[v] < \infty$ and does A)3).

C). If $b[v] < \infty, b[v]$ is increased by 1 for every MLD and does A)3).

Table 10: Routing states of an arbitrary node v

Node state	Route entry format	Domain of Value	Meaning
No routing state (NS)	(∞, nil, ∞)	None	Initial state
Formative state (FS)	$(d[v], p[v], \infty)$	$0 \leq d[v] < \infty$	Valid route
Broken state (BS)	$(\infty, nil, b[v])$	$0 \leq b[v] < \infty$	Broken route
Wait state (WS)	$(d[v], p[v], b[v])$	$0 < b[v] < d[v] < \infty$	Waiting

D). When a link to node u is formed. If $d[v] < \infty$ and $b[v] = \infty$, sends a PU message $(d[v], p[v])$ to node u.

E). When a link (v, u) fails. If $p[v] = u$, broadcasts a NU message $(d[v], p[v])$ and sets $b[v] = d[v]$, $d[v] = \infty$ and $p[v] = nil$.

Algorithm A) and B) processes the *PU* message and the *NU* message. The condition $p[u] \neq v$ guarantees that the parent of the sender will not process the message. Algorithm A)1) updates the route entry of node v. It updates the values only if node v can gain a smaller tree level. A)2) decides if a *PU* message needs to be sent. A node sends a *PU* message only if it is in the formative state and its tree level is changed (decreased) by A)1). A)3) tests its safe reply condition (*SRC*) when node v is in the wait state to figure out if it can enter the formative state and send a *PU* message. Algorithm B)1) cleans up the invalid route entry if the sender is the parent of node v, which is in formative state or wait state. If node v receives a smaller broken tree level, it also updates its broken tree level. B)2) decides if a *NU* message needs to be sent. A node sends a *NU* message only if it changes its state from the formative state or waiting state to the broken state. B)3) tests its *SRC* as A)3) does. Algorithm A), B) are illustrated in Fig. 51 and 52. Fig. 53 shows the finite state graph of our routing algorithm A) and B). Algorithm C), D) and E) process the events of time-out, addition of link and link failure, which are not showed in the finite state graph.

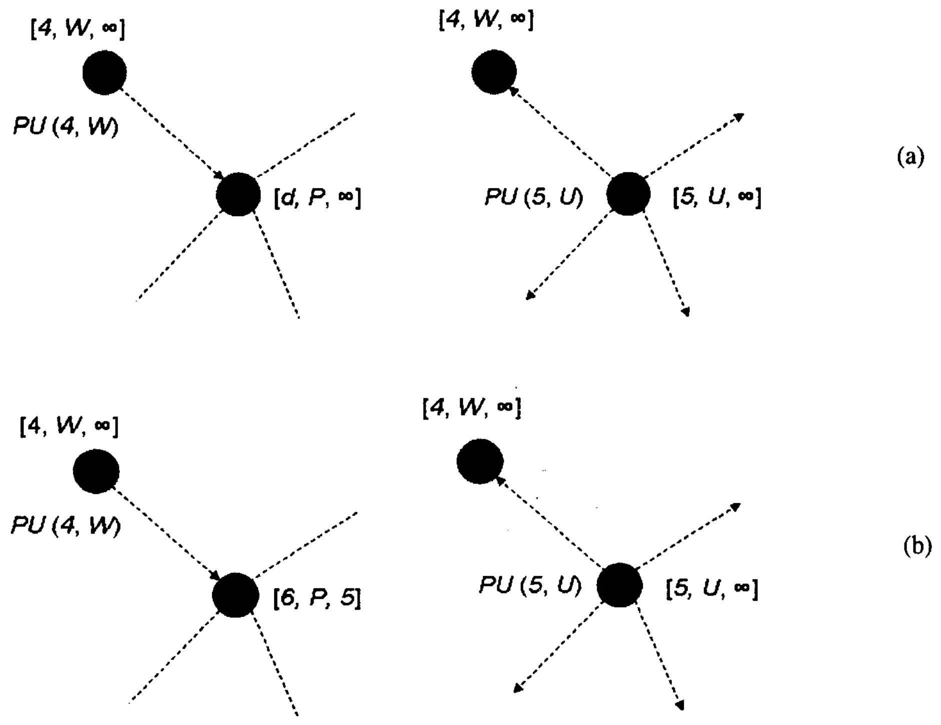


Fig. 51: Illustrating algorithm A. (a) Receiving a *PU* message when node v in state NS or PS. Only send a *PU* message if d is decreased (d will never be increased by receiving a *PU* message. (b) Receiving a *PU* message when node v in state BS or WS. Only send a *PU* message if $SRC = d - b \leq 0$.

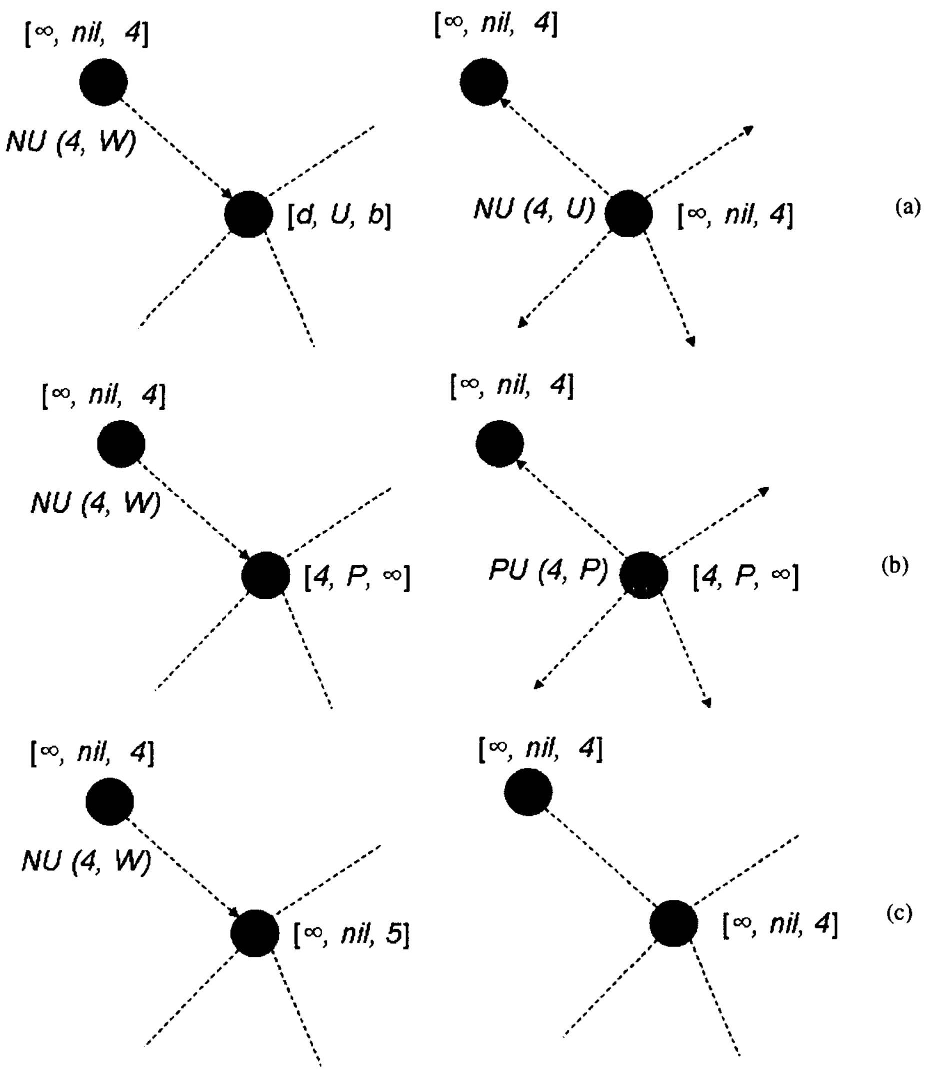


Fig. 52: Illustrating algorithm B. (a) Receiving a *NU* message when node v in state FS or WS, Only send a *NU* message if d is set to ∞. (b) Receiving a *NU* message when node v in state FS. Only send a *PU* message if $SRC = d - b \leq 0$. (c) Receiving a *NU* message when node v in state BS. Only update b if b is decreased. No message is sent.

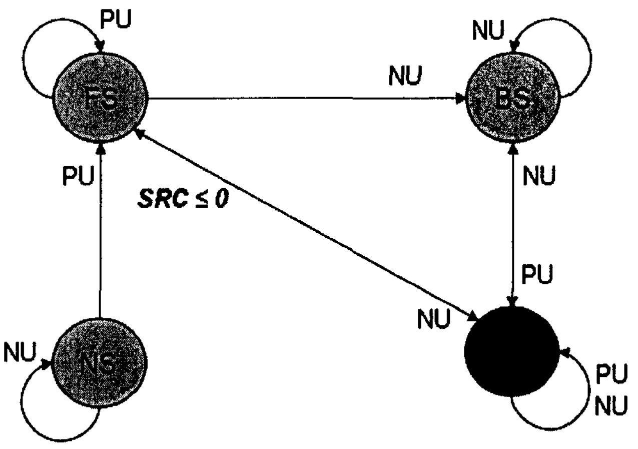


Fig. 53: Finite state machine of routing protocol.

VI.5 PROOF OF CORRECTNESS

Definition VI.1 (Route loop) *A path from node d to node s is a sequence of nodes in which $(d, e), \cdots, (m, n), \cdots, (r, s)$ are links in the path. A route from node d to node s is a path, in which node $d, e, \cdots, m, n, \cdots, t$, s maintain a set of parent pointer, where $p[d] = e, \cdots, p[m] = n, \cdots, p[t] = s$. A route loop is a route in which a node v appears more than once.*

Lemma VI.2 *No route loop will be formed when a node v updates its tree level $d[v]$ as a result of receiving a control message from its neighbor node u.*

Proof.

This can be proved by contradiction. Assume node v receives a message and updates its tree level $d[v]$, a route loop including node v occurs for the first time. A node can only receive two kinds of message: *PU* message and *NU* message. Assume node v receives a *NU* message, if node v updates $d[v] = d[u]$ where $d[u] < \infty$(Algorithm B1), the previous route to the sink is erased. No path, no loop. Assume node v receives a *PU* message, if node v updates $d[v] = d[u] + 1$, a route loop including node v occurs for the first time. The path from v to s must include v.

Assume the *PU* message is from node u. Say $d[v]$ is the tree level before node v updates it. Then $d[v]$ must be smaller than $d[u]$ since node v is a down-tree node of node u. Therefore $d[v] < d[u] + 1 = d[v]$. However, node v will only update $d[v]$ if $d[v] > d[u] + 1$. It is a contradiction. □

Theorem VI.3 *No route loop will be formed at every instant of time*

Proof. It follows immediately from Lemma VI.2, and note that a tree link failure, a link addition and time out event may generate control messages but they will not form a route loop. □

Lemma VI.4 *If node v receives a* NU *message $(b, *)$ from node u, which changes its state from WS or FS $(d[v], u, *)$ to BS $(\infty, *, b)$, then $b < d[v]$.*

Proof. A *NU* message $(b, p[w])$ is initialized at the broken tree level $b = d[w]$ and propagated towards up-tree of node w, where each node v has a tree level $d[v] > d[p[v]] > \cdots > d[w] = b$. □

Definition VI.5 (b-propagation cycle) *If a* NU *$(b, *)$ message is propagated in a cycle, the cycle is called a b-propagation cycle.*

Lemma VI.6 *Consider a finite network in which an arbitrary but finite series of topology changes occur between* 0 *and time t; no changes occur after time t. If node v sends* NU *messages that contains $(b, *)$ infinite times, a b-propagation cycle must exist in the network.*

Proof. Since a finite series of topology changes occur between 0 and time t, only finite tree links failed between 0 and time t. And a broken tree level b never changes its value during its propagation. We assume the broken tree level set (**B-Set**) is $(b_0, b_1, \cdots, b_i, \cdots, b_{n-1}, b_n)$ where $b_0 \leq b_1 \leq \cdots \leq b_i \leq \cdots \leq b_{n-1} \leq b_n$. Since the **B-Set** is a finite set, one of the broken tree levels must be propagated along within a *NU* message, which will visit node v infinite times.

Assume node v sends a *NU* message contains broken tree level b infinite times after time t, where $b \in$ **B-Set**. If no b-propagation cycle exists in the network, the *NU* message sent from node v will not visit node v again. Therefore one of its neighbors u must send a *NU* $(b, *)$ message infinite times and the message will not revisit node v and u. And a neighbor w of node u also must send a *NU* $(b, *)$ message infinite times and the message will not revisit node v, u and w, and so on, which implies that node v sends a *NU* $(b, *)$ message infinite times only if the network has infinite number of nodes. It is a contradiction. □

Corollary VI.7 *The time for a* NU *$(b, *)$ message to revisit a node on the b-propagation cycle is not greater than $k * MLD$, where k is the length of the cycle.*

Proof. In order for the *NU* message propagating on the cycle, the *NU* message has to be sent by each node on the cycle. The time for the *NU* message to be received by the next node on the cycle will be less then *MLD* (by definition). The total time for the *NU* message to travel is less than $k * MLD$. □

Lemma VI.8 *Consider a finite size network in which an arbitrary but finite series of topology changes occur between 0 and time t; no changes occur after time t. Then a finite time after t, all message activity will be cease.*

Proof.

Case 1: Assume node w sends *NU* messages infinite times after time t.

Because the **B-Set** is a finite set, node w must send *NU* messages contain $(b, *)$ infinite times, where $b \in$ **B-Set**. Based on Lemma VI.6, a b-propagation cycle must exist in the network (Fig. 54).

Assume a b-propagation cycle consists of a set of nodes $(v^1, v^2, \cdots, v^i, \cdots, v^k)$ and a *NU* message $(b, *)$ is propagated on the cycle in that order and revisits node v^1 after it visits node v^k. According to the algorithm B)1) and B)2), node v^i propagates a *NU* message received from $p[v^i]$ only when it is in FS or WS, where $p[v^i] = v^{i-1}$ or $p[v^i] = v^k$ if $i = 1$ and its tree level $d[v^i] = d[p[v^i]] + 1$. In order to enter the FS or WS, node v^i must receive a *PU* message from $p[v^i]$ and set its new tree level $d[v^i] = d[p[v^i]] + 1$. In other words, a *NU* message sent by node v^i only propagates to its up-tree nodes. After the *NU* message cleans up the tree level $d[v^i]$, the tree levels of its children will not be changed or the *NU* message will stop. Hence, when the *NU* message revisits node v^i, its tree level must be $d[v^i] + k$, where k is the cycle length. A *PU* message must be propagated in the cycle ahead of the *NU* message and the *NU* message will not catch up the *PU* message and vise versa. The *PU* message must contain a tree level which is increase by 1 at each node on the cycle.

As illustrated in Fig. 55, assume node v^1 changes its route entry from $(d, *, *)$ to (∞, nil, b) and sends a *NU* message $(b, *)$ at time $t^1 > t$. A *PU* $(d, *)$ message

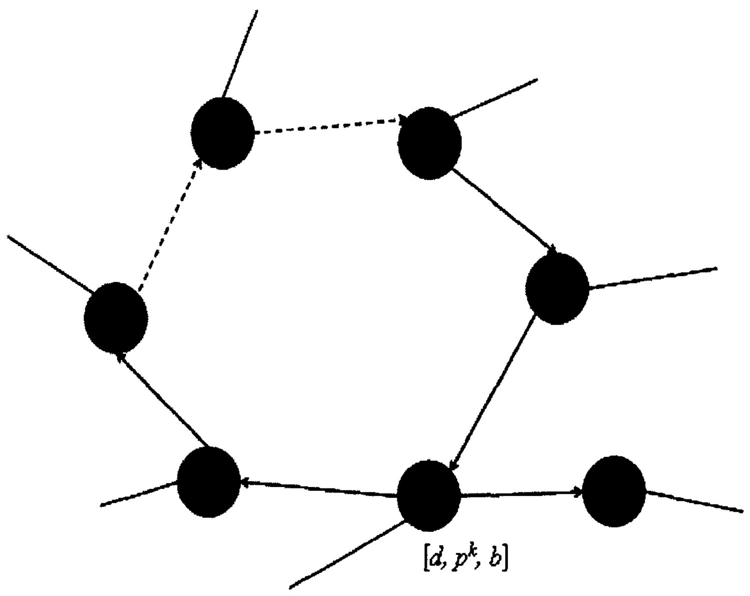


Fig. 54: Illustrating a b-propagation cycle.

must be sent by node v^1 at some time $t^0 < t^1$. Also assume node v^1 receives the propagated *PU* message $(d + k - 1, v^{k-1})$ from node v^k at time $t^k > t^1$, sends a *PU* message $(d + k, p^k)$ at time $t^w \leq t^k$ and sends a *NU* message $(b, *)$ again at time $t^z > t^w$. If no control message is received between time t^1 and time t^k, the route entry is $(\infty, nil, b + MC)$ at time t^k, where

$$MC = \left\lfloor \frac{t^w - t^1}{MLD} \right\rfloor \tag{47}$$

The *SRC* must be satisfied at time t^w for sending the *PU* message, we have:

$$d + k - b - MC \leq 0 \tag{48}$$

Base on Corollary VI.7, the time needed for a *NU* message to revisit a node on the cycle must not be greater than $k * MLD$ ($t^z - t^1 \leq k * MLD$), so we have:

$$MC < k \tag{49}$$

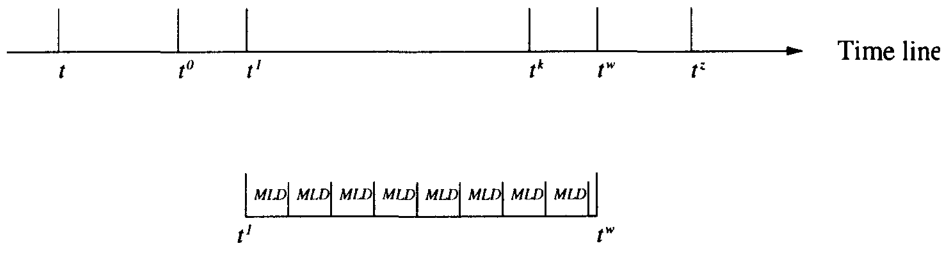


Fig. 55: Illustrating that a node never sends the same *NU* message twice.

From equations (48) and (49) we have:

$$d < b \tag{50}$$

But from Lemma VI.4, we have

$$d > b \tag{51}$$

It is a contradiction. Hence, some control message must be received between time t^1 and time t^k, which modifies the route entry.

We divide the time period of $t^k - t^1$ into $MC + 1$ slots. The length of each slot (except the last slot) is MLD (See Fig. 55). We index the slots as $0, 1, \cdots, i, \cdots, MC$.

Based on the Algorithm B (the algorithm only updates b if b decreases), in a slot i, the value of $b[v^1]$ will not be larger than $b + i$ or $b[v^1] = \infty$. If $b[v^1] = \infty$, then we have $d[v^1] \leq b + i$ (The value of b is only sets to ∞ if the condition SRC is satisfied. Hence, during MC slot, the route entry of node v^1 can be in one of three forms:

1. In FS state $(d[v^1], *, \infty)$, where $d[v^1] \leq b + MC$
2. In BS state $(\infty, nil, b[v^1])$, where $b[v^1] \leq b + MC$
3. In WS state $(d[v^1], *, b[v^1])$, where $b[v^1] \leq b + MC$

If node v^1 is in FS state at time t^k when it receives the *PU* message $(d+k-1, v^{k-1})$, $d[v^1]$ will not be updated since $b + MC \leq d + k$. And when the *NU* message arrives at time t^z, node v^1 will not send the *NU* message. Hence, the b-propagation cycle is broken at node v^1.

If node v^1 is in BS or WS state, we already showed that $d < b$ must be satisfied and it is a contradiction.

We conclude that no b-propagation cycle exists in the network. Hence node w can not send *NU* messages infinite times after time t.

Actually, we proved that no b-propagation cycle can be formed at any moment of time. In other word, no node will send the same *NU* message twice.

Case 2: Assume that node w sends *PU* messages infinite times after time t.

In case 1, we proved that a node will not send *NU* messages infinite times. Therefore a node has to receive *PU* messages infinite times in order to keep generating new *PU* messages. Based on A)1), 2), a node will send a new *PU* message only if its tree level is decreased by receiving a *PU* message or a *PU* message is generates by topology changes. Since a tree level value in a *PU* message is smaller than ∞, we conclude that node w can not send *PU* messages infinite times after time t. □

Lemma VI.9 *If and when all messages activity has ceased, all tree levels in all the routing entry will be correct.*

Proof. Assume node v has an incorrect route entry after all messages activity has ceased. Case 1: $d[v] = 1$. Node v is a neighbor of the sink and it knows the sink is its neighbor, $d[v]$ must be correct. Case 2: $1 < d[v] < \infty$ This can be proved by contradiction. Assume node v is the lowest level tree node has an incorrect routing entry $(d[v], p[v])$, then node $p[v]$ must also have an incorrect routing entry where $d[p[v]] < d[v]$, and it contradicts the assumption that node v is the lowest level tree node. Case 3: $d[v] = \infty$ If there is not route between node v and the sink s, then $d[v]$ is correct. Otherwise all nodes in that route must have $d[x] = \infty$. Hence the second last node n on the route must have $d[n] = \infty$, too. But the neighbors of the sink have $d[n] = 1$. It is a contradiction. □

Theorem VI.10 *Consider a network in which an arbitrary but finite series of topology changes occur between 0 and time t; no changes occur after time t. Then a finite time after t, all distance values in all the routing entry must be correct.*

Proof.

It follows immediately from Lemma VI.8 and Lemma VI.9. □

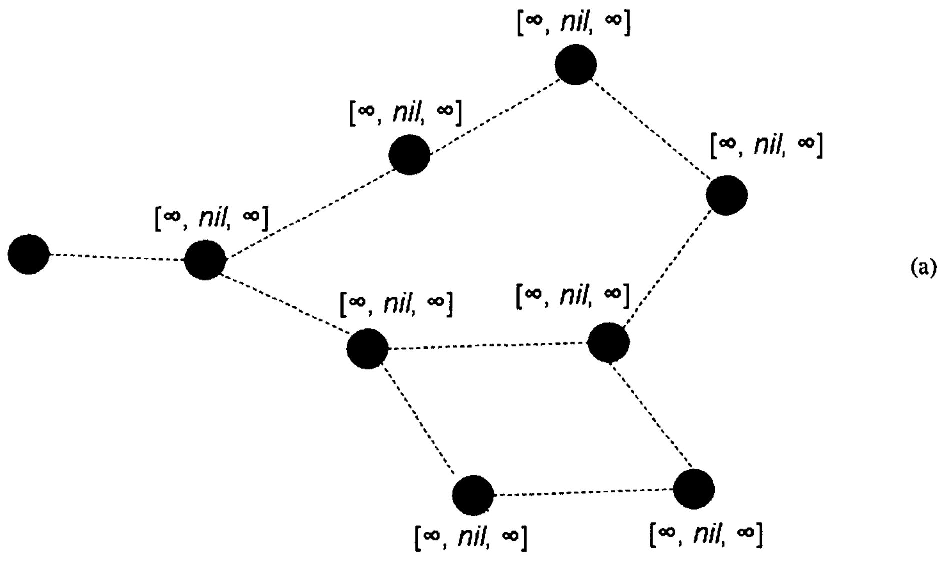


Fig. 56: Route construction(a).

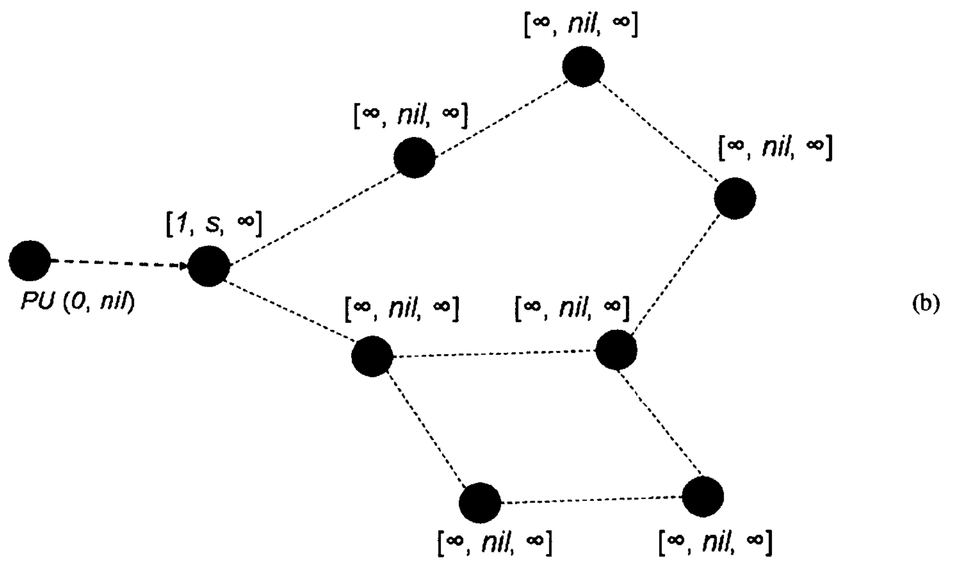
VI.6 EXAMPLES

In this section, we present examples of the way in which our routing protocol operates on various situations, such as creating the shortest path tree, erasing invalid routes and rebuilding the tree after link failures.

Fig. 56 (a) depicts a small network with eight nodes with a sink on the left. Figs. 57, 58 and 59 show how the route entry changes at each node during the shortest path tree creation.

Figs. 60, 61 and 62 show the *NU* message propagation after the link (S, A) fails. In Figs. 61 and 62, the *NU* message propagation is delayed at node E and D, respectively. The delay at node D cause node H stays in state WS. Eventually, a *NU* message is received (Fig. 62) and the route information at node H is updated.

Figs. 63, 64, 65, 66, and 67 show how the tree is recovered after the link (A, C) fails. Since the route at node E is not effected by the link fail, after the $SRC \leq 0$, node E enters the state FS and sends a *PU* message. The *PU* message is forwarded at each node in the failure up-tree and a new shortest path tree is reconstructed.



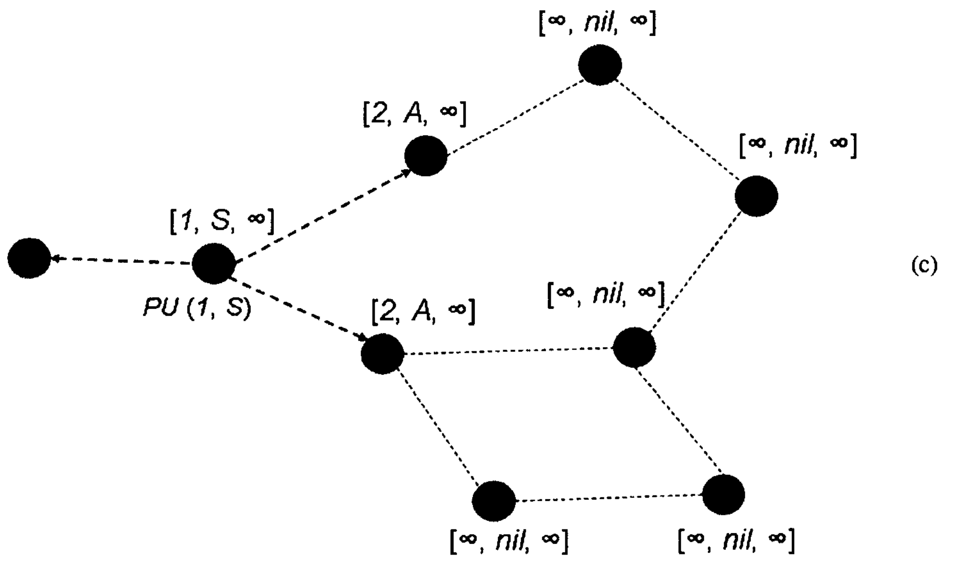
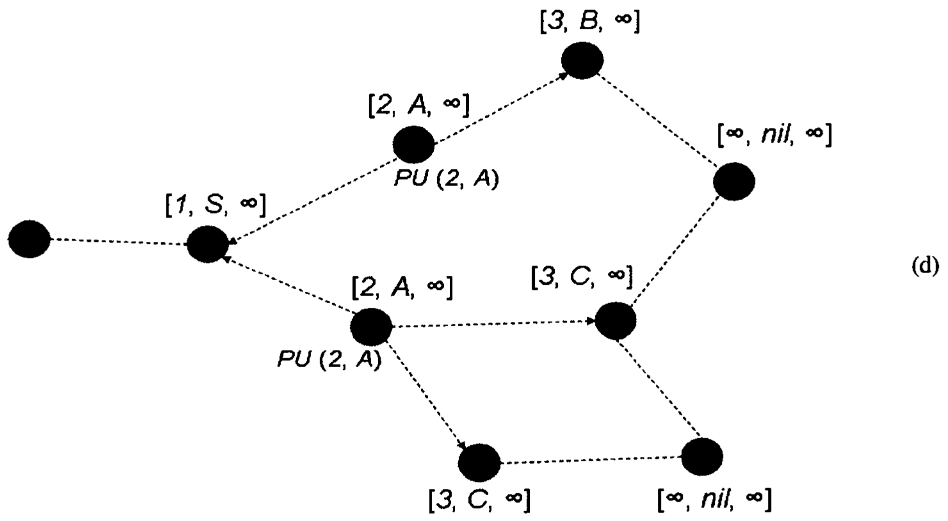


Fig. 57: Route construction(b, c).



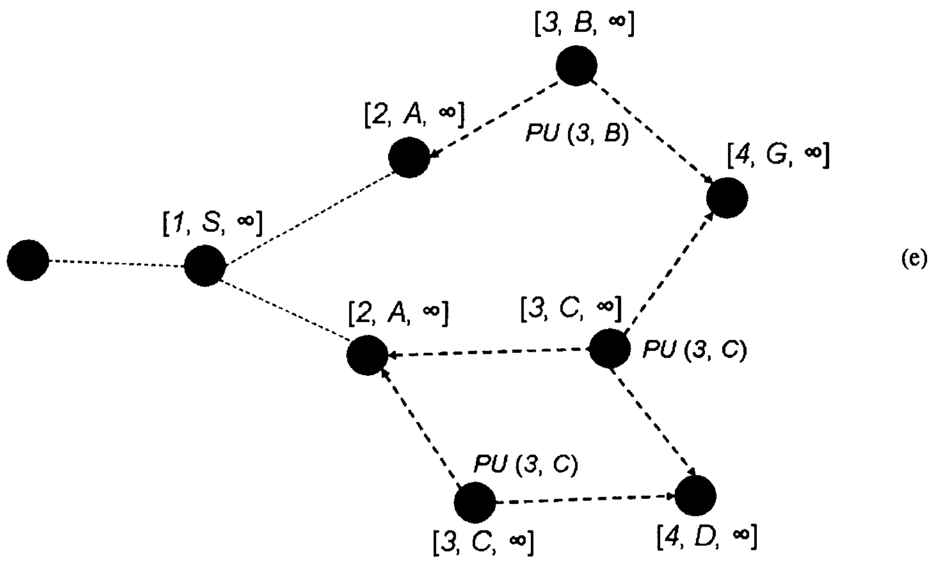
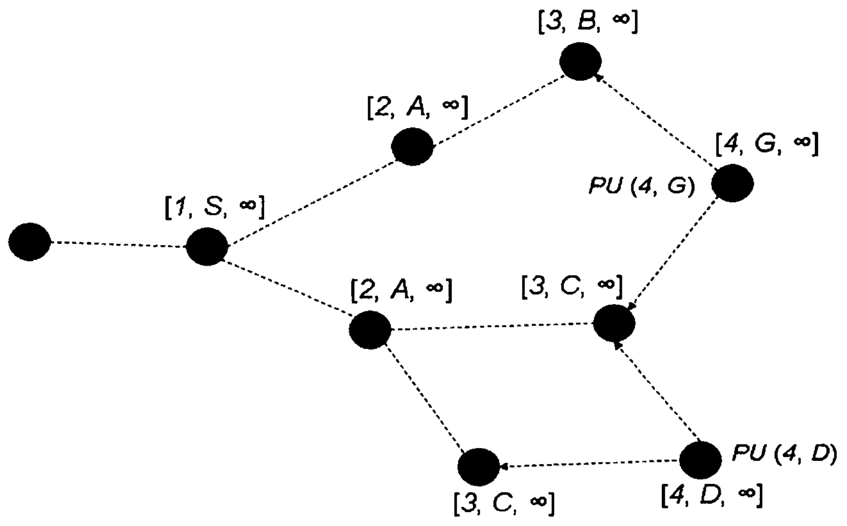
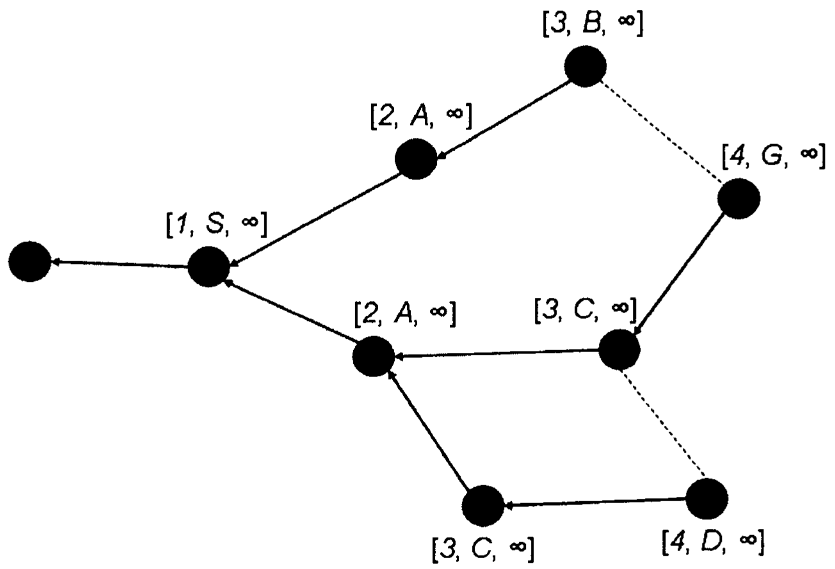


Fig. 58: Route construction(d, e).

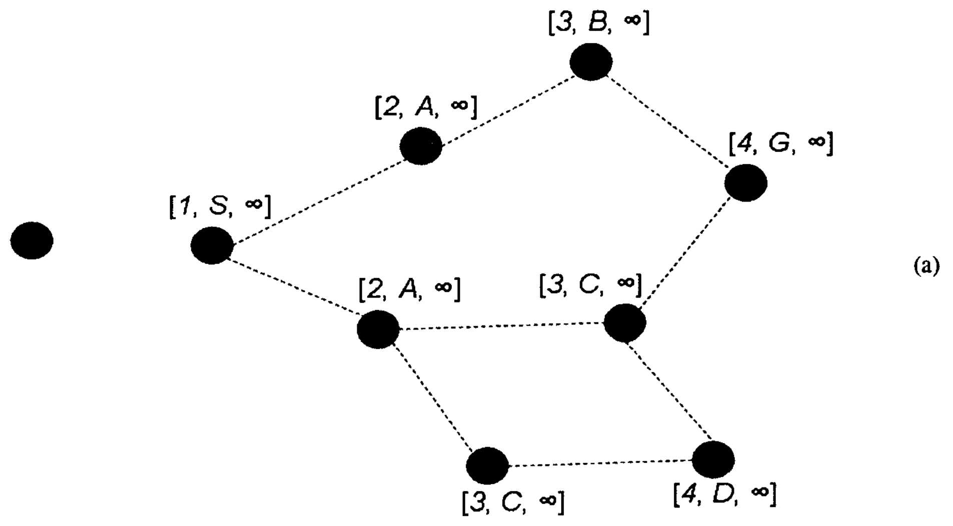


(f)



(g)

Fig. 59: Route construction(f, g).



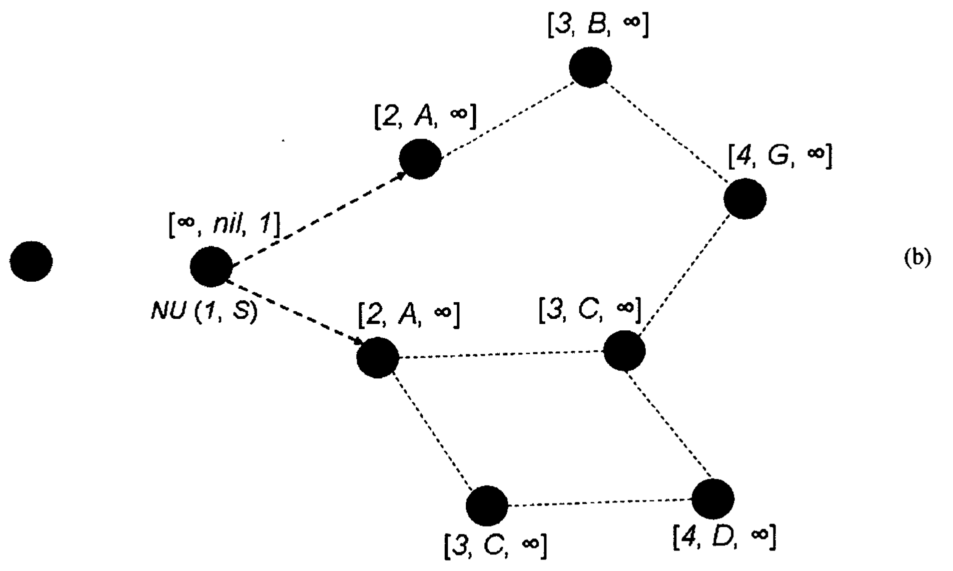
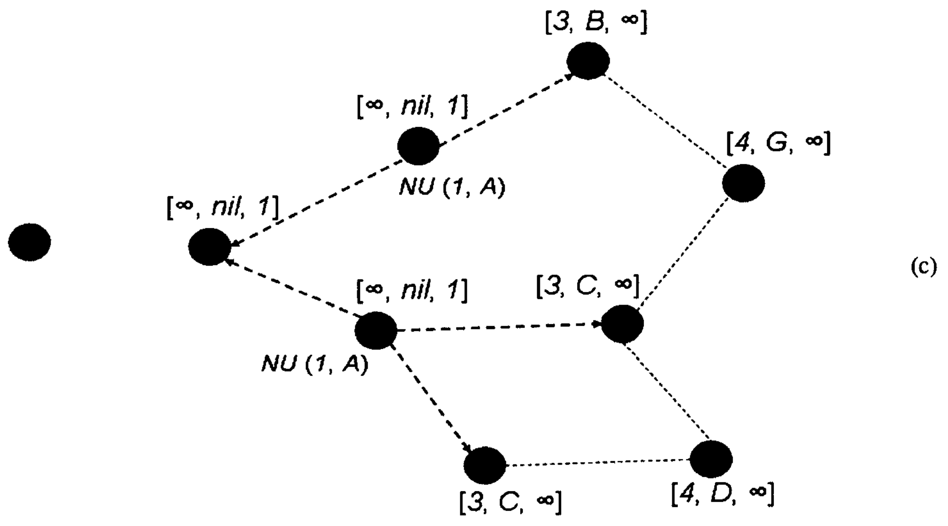


Fig. 60: Invalid route erasure(a, b).



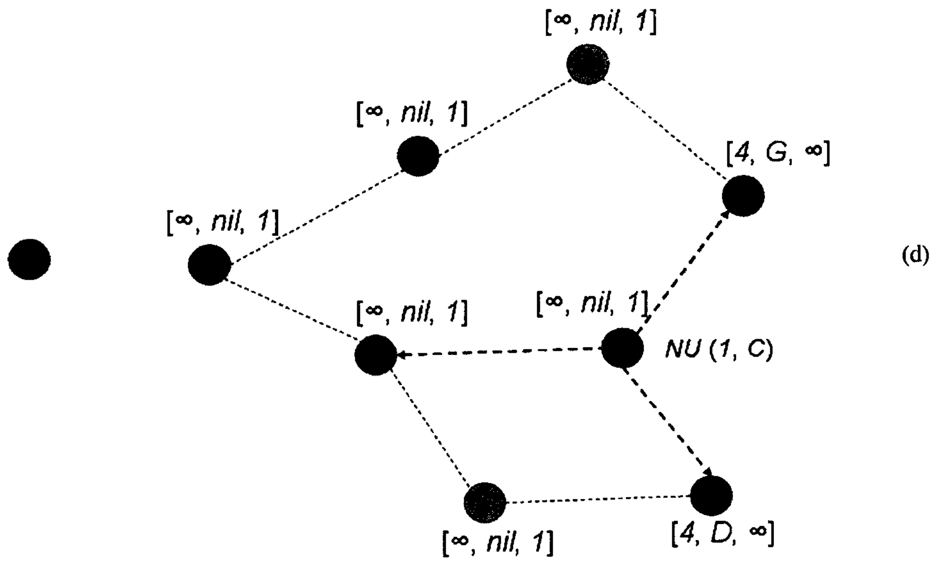
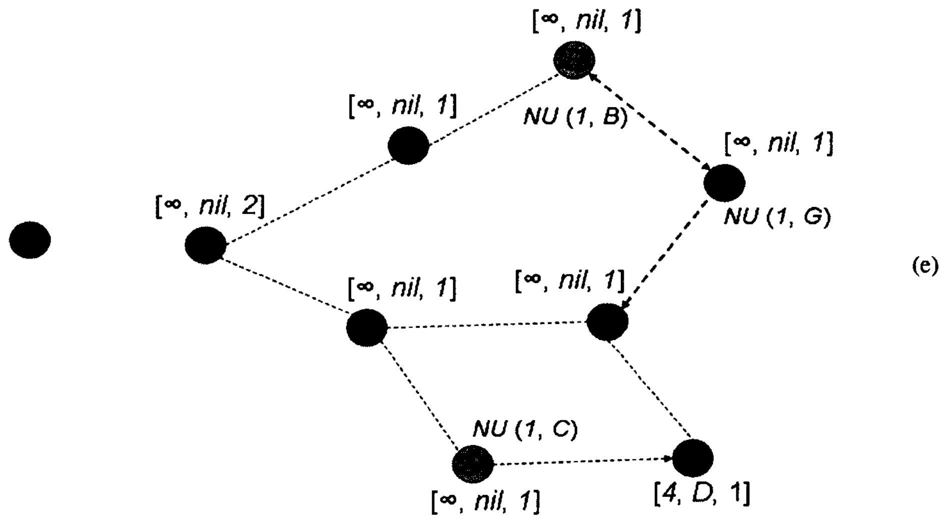


Fig. 61: Invalid route erasure(c, d).



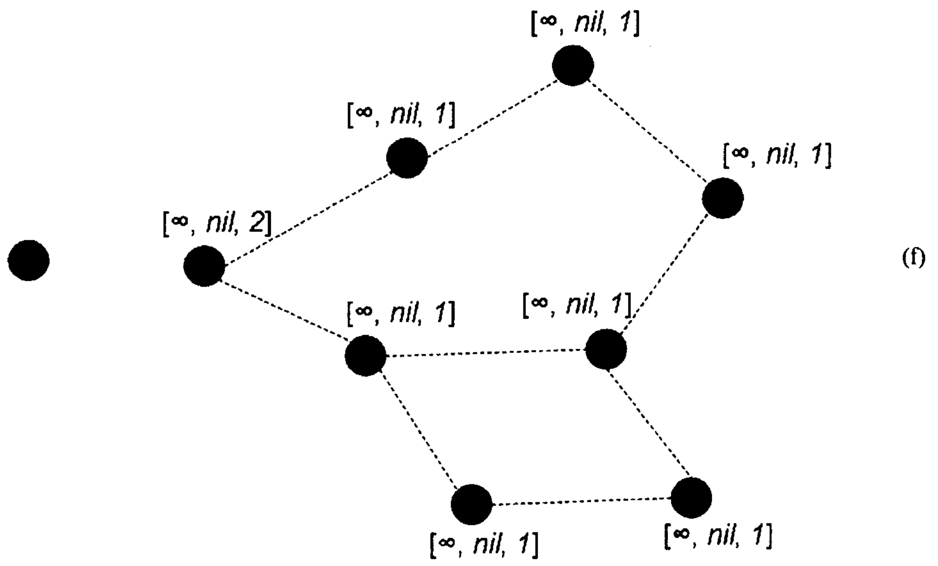
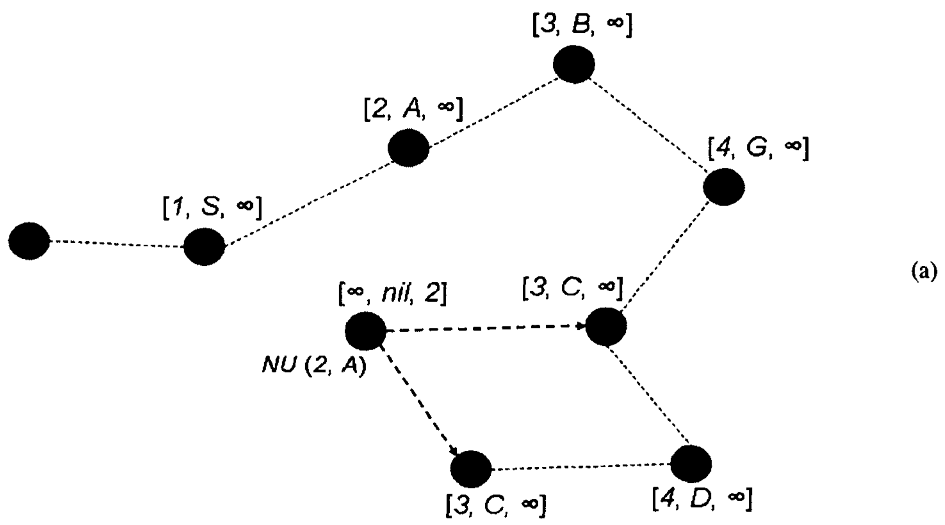


Fig. 62: Invalid route erasure(e, f).



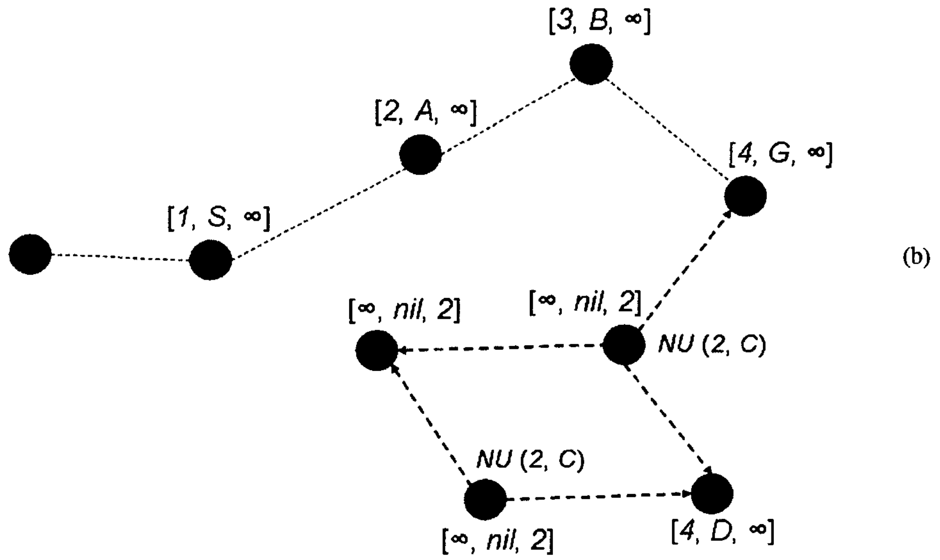
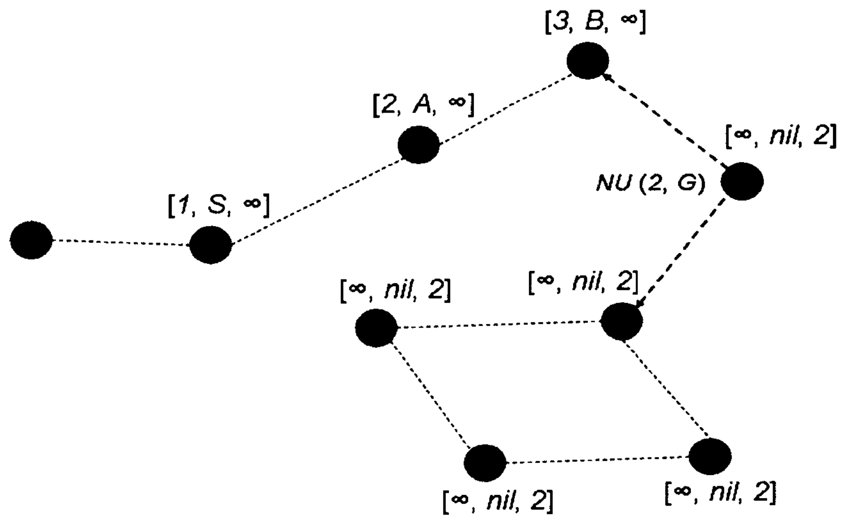
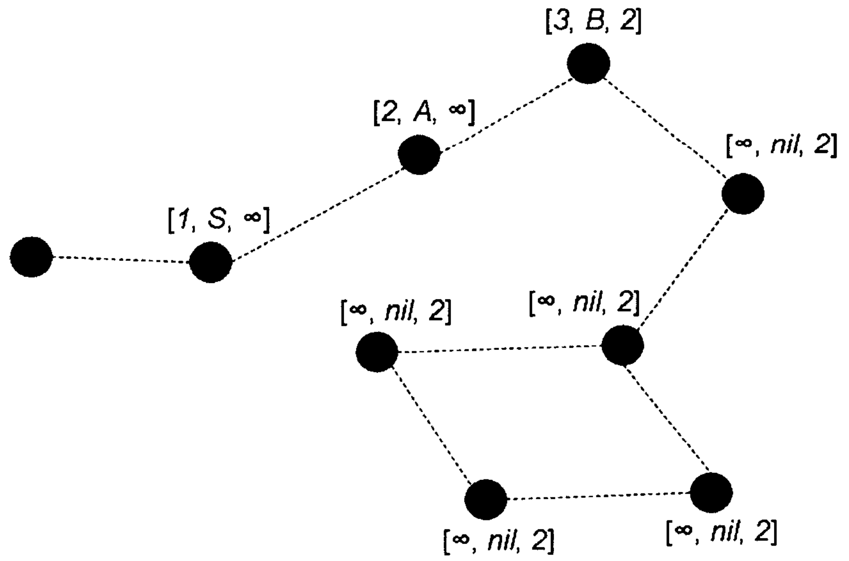


Fig. 63: Route reconstruction after a link failure(a, b).

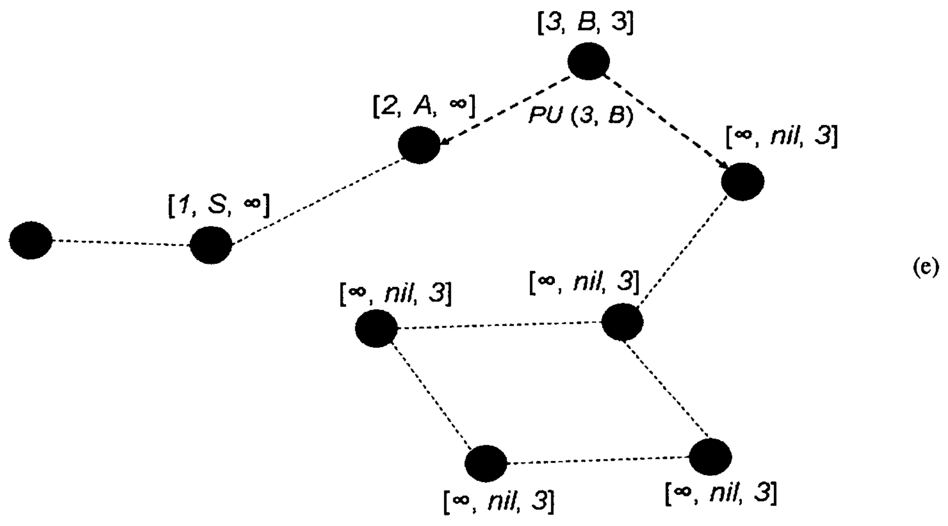


(c)



(d)

Fig. 64: Route reconstruction after a link failure(c, d).



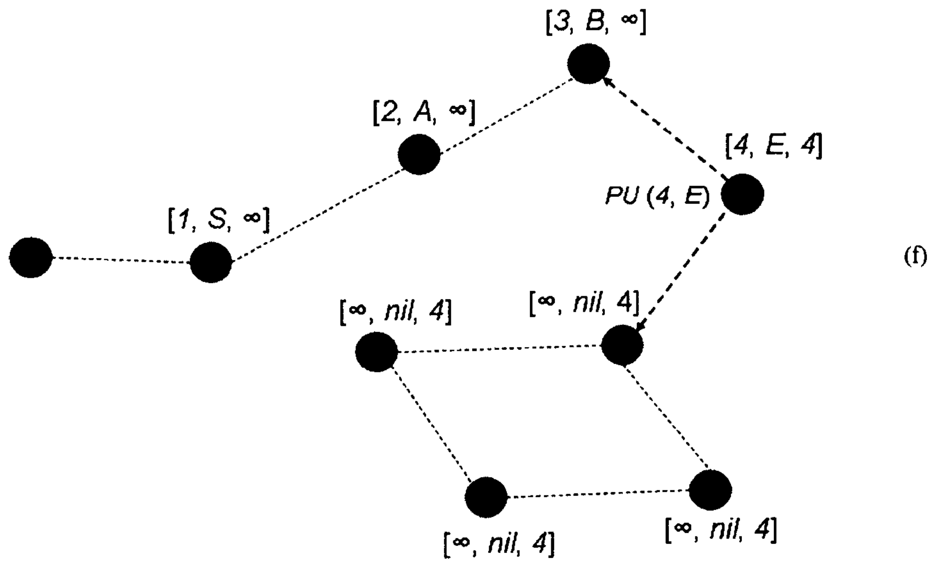
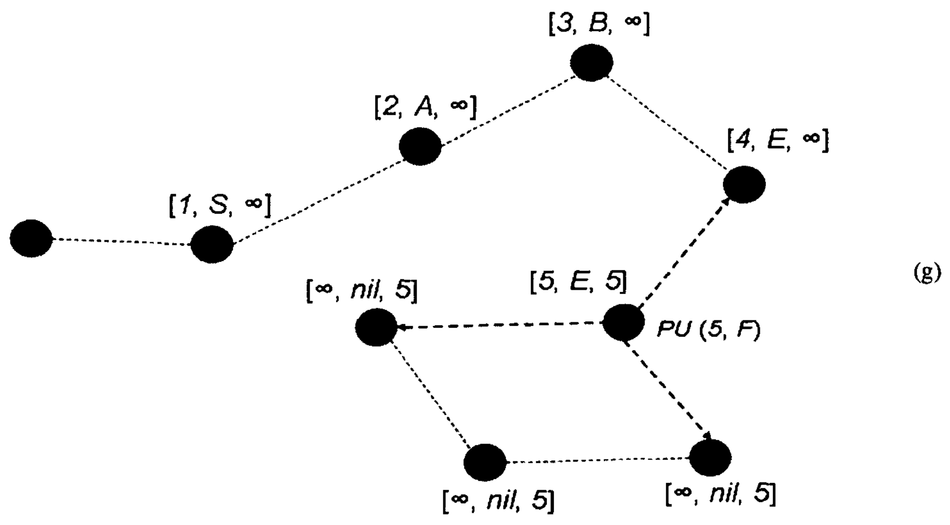


Fig. 65: Route reconstruction after a link failure(e, f).



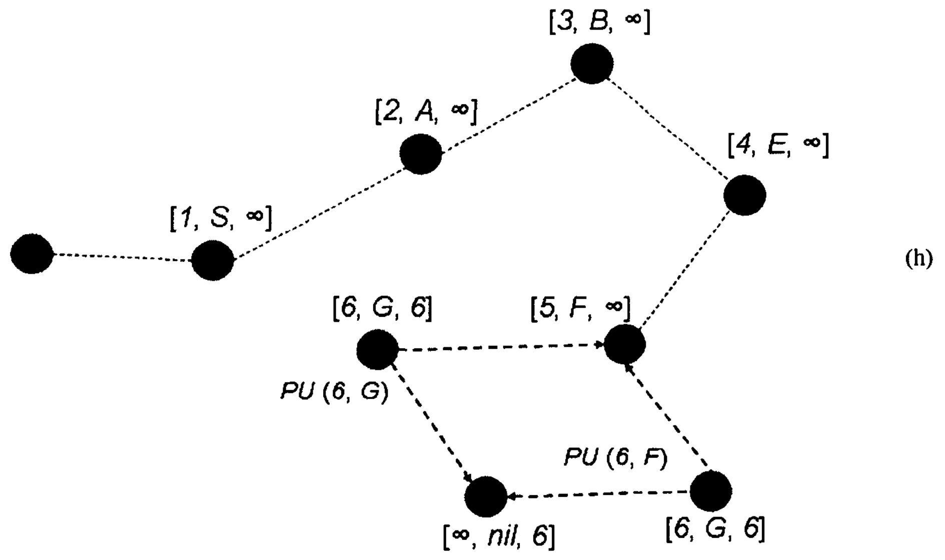
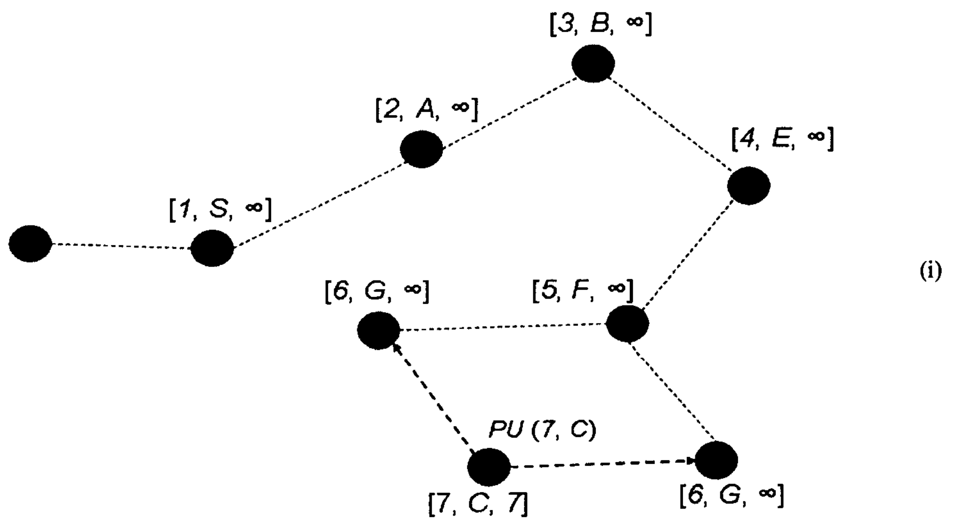


Fig. 66: Route reconstruction after a link failure(g, h).



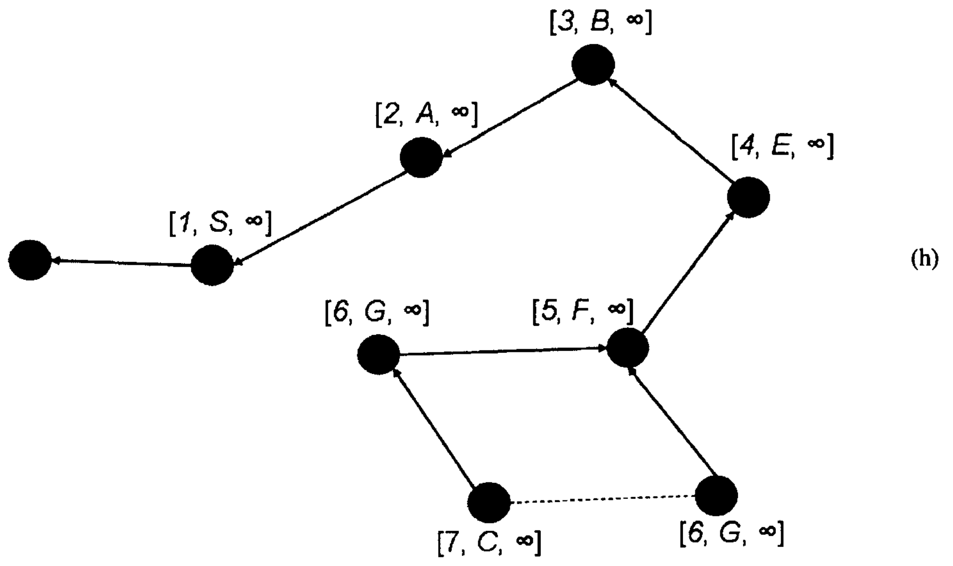


Fig. 67: Route reconstruction after a link failure(i, j).

VI.7 PERFORMANCE ANALYSIS

In this section we compare the worst case failure recovery performance of the algorithm presented in Section VI.4 to the decomposed DVP and the algorithms proposed in [65] and [52]. The comparison is made in terms of the time and communication overhead required by the various algorithms to converge to correct route entries. For the comparison, we assume that all the algorithms behave synchronously, so that every node in the network executes a "step" of the algorithm simultaneously at fixed points in time. At each step, a node processes a single input event (a message from a neighbor node or a time out). We also assume every control message is sent within the maximum delay of MLD. Hypothetically, we can think that the time is divided into equal length (MLD) of slots. Every node sends and receives the control messages or processes the time out event at the slot boundary.

Based the above assumption, we compare the performance of the various algorithms in terms of the *time complexity* and the *communication complexity*, two terms we borrowed from [27]:

> The performance of a routing algorithm is quantified in terms of the number of steps called time complexity or TC and the number of messages called communication complexity or CC, required by each algorithm after a single link failure.

In the worst case, the DVP has $TC = N, CC = N^2$ [27], where N is the number of network nodes, for a single link failure. The Merlin-Segall algorithm has $TC = 2d, CC = N^2$ [27], where d is the network diameter. The JAFFE-MOSS algorithm 1 ([52]) have $TC = x, CC = 2x$, where x is the number of nodes in the failure up-tree. The JAFFE-MOSS algorithm 2 has $TC = 3h_1 + h_2, CC = 4x$, where h_1 is the height of failure up-tree and h_2 is the height of the recovered up-tree. Our routing algorithm has $TC = h_1 + h_2, CC = 2x$, since, to reconstruct the shortest path tree, only two messages are propagated on affected sub-tree, a NU message to erase the invalid route entry and a PU message to set the new route entry. Particularly, if the link failure leaving the network disconnected, our algorithm is extremely efficient: only a PU message is needed to be propagated on the failure up-tree. In this case, $TC = h_1, CC = x$.

It is clear from the above results that our routing algorithm has a better performance in terms of communication overhead. Table 11 summaries the performance

Table 11: Complexity comparison

Algorithms	TC	CC
DVP	$N = O(N)$	$N^2 = O(N^2)$
Merlin-Segal	$2d = O(d)$	$N^2 = O(N^2)$
JAFFE-MOSS algorithm 1	$x = O(x)$	$2x = O(x)$
JAFFE-MOSS algorithm 2	$3h_1 + h_2 = O(h)$	$4x = O(x)$
New Algorithm	$h_1 + h_2 = O(h)$	$2x = O(x)$

comparison, where h is the diameter of the failure up-tree.

CHAPTER VII
CONCLUSIONS

This dissertation has explored the principles and methodologies of the sensor network protocol design. The requirement analysis makes it clear that one of the most important requirements is to prolong network lifetime. We believe that using localized algorithms and constructing virtual communication infrastructure are the keys to meet this requirement.

The self-organization protocol presented in Chapter IV is the backbone of the dissertation. The protocol is developed (using localized algorithms) to construct and maintain a communication infrastructure, which supports the development of many energy efficient protocols for local scheduling, routing and training.

In Chapter V and VI, we presented the training and routing protocols, which, particularly, benefit from the communication infrastructure. The simulation and analysis showed that the energy efficiency is achievable.

VII.1 FUTURE RESEARCH DIRECTIONS

There are many research questions raised by the research, and many other which are important and have not yet been addressed. We list several directions of the future research:

1. The effect of the communication infrastructure at the application level.

   The application specific nature of the sensor network requires data centric protocols. The application information can improve the efficiency of the low level communication protocols. However, there will not be a general solution for that matter. Although the communication infrastructure provides a platform to design such protocols, the design has to be based on the knowledge of the particular applications as well.

2. The effect of the toggling between sleep and wake periods.

   The physical characteristics of the sensor network can dramatically influence the protocol design. In order to achieve energy efficiency, we need to know the power consumptions of the individual sensors. In particular, how many energy is wasted by switching the radio on or off? This knowledge can be critical for prolonging network lifetime. For instance, in a beacon frame, should the protocol let the leader listen for a specific slot or the entire frame? Another example would be how to choose an optimized protocol for location training. To answer those questions, further research on energy consumption characteristic is required.

3. The effect of the various data traffic on the data routing protocol.

   The performance of a routing protocol can be significantly different for various data traffic. It is more so than ever in the sensor network, where the data traffic could follow some patterns depending on the applications. Again, this reflects the application specific nature of the sensor network and has to be studied case by case.

REFERENCES

[1] J. Agre, L. Clare, An integrated architecture for cooperative sensing networks, IEEE Computer 33(5) (2000) 106-108.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramanian, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38(4) (2002) 393-422.

[3] J. Albowicz, A. Chen, L. Zhang, Recursive position estimation in sensor networks, Proceedings of the Ninth International Conference on Network Protocols(ICNP'01) Washington, DC, November 2001.

[4] D. J. Baker, A. Ephremides, J. A. Flynn. The design and simulation of a mobile radio network with distributed control, IEEE Journal on Selected Areas in Communications 2(1) (1984) 226-237.

[5] S. Bandyopadhyay, E. Coyle, An efficient hierarchical clustering algorithm for wireless sensor networks, The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM'03), San Francisco, CA, April 2003.

[6] M. S. Blumenthal, D. D. Clark, Rethinking the design of the Internet: the end to end arguments vs. the brave new world. ACM Transactions on Internet Technology 1(1) (2001) 70-109.

[7] D. Braginsky, D. Estrin, Rumor routing algorithm for sensor networks, Proceedings of Workshop on Sensor Networks and Applications(WSNA), October 2002.

[8] J. Carle, D. Simplot-Ryl, Energy-efficient area monitoring by sensor networks, IEEE Computer, February 2004, 40-47.

[9] I. Chatzigiannakis, S. Nikoletseas, A sleepawake protocol for information propagation in smart dust networks, Proceedings of IEEE International Parallel and Distributed Processing Symposium, Nice, France, April 2003.

[10] J.-H. Chang, L. Tassiulas, Routing for maximum system lifetime in wireless adhoc network. Proceedings of 37th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, September 1999.

[11] J.-H. Chang, L. Tassiulas, Energy conserving routing in Wireless Ad-hoc Network, The 19th Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM'00), Tel Aviv, Israel, March 2000.

[12] H. Chan, A. Perrig, ACE: an emergent algorithm for highly uniform cluster formation, Proceedings of 1st European Workshop on Sensor Networks, Berlin, Germany, January 2004, 154-171.

[13] D. D. Clark, The design philosophy of the DARPA Internet protocols, The Conference of the Special Interest Group on Data Communication(SIGCOMM), August 1988, 106-114.

[14] D. Culler, D. Estrin, M. Srivastava, Overview of sensor networks, IEEE Computer 37(8) (2004) 41-49.

[15] E. W. Dijkstra, C. S. Scholten, Termination detection for diffusing computations, Information Processing Letters 11(1) (1980) 1-4.

[16] J. Ding, K. M. Sivalingam, R. Kashyapa, L. J. Chuan, A multi-layered architecture and protocols for large-scale wireless sensor networks, Proceedings of the IEEE Vehicular Technology Conference(VTC'03), Orlando, FL, October 2003.

[17] L. Doherty, K. S. J. Pister, L. E. Ghaoui, Convex position estimation in wireless sensor network, The 20th Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM'01), Anchorage, AL, April 2001.

[18] D. M. Doolin, S. D. Glaser, N. Sitar. Software architecture for GPS-enabled wildfire sensorboard, TinyOS Technology Exchange, University of California, Berkeley CA, February 2004.

[19] D. M. Doolin, N. Sitar, Wireless sensors for wildfire monitoring, Proceedings of SPIE Symposium on Smart Structures & Materials (NDE'05), San Diego, CA, March 2005.

[20] J. Elson, D. Estrin, Time synchronization for wireless sensor networks, Proceedings of the 2001 International Parallel and Distributed Processing Symposium(IPDPS), April 2001.

[21] C. C. Enz, A. El-Hoiydi, J.-D. Decotignie, V. Peiris, WiseNET: an ultralow power wireless sensor network solution, IEEE Computer 37(8) (2004) 62-69.

[22] http://www.stetson.edu/efriedma/cirincir

[23] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: scalable coordination in sensor networks, Proceedigns of the ACM Mobile Computing and Networking(MOBICOM'99), Seattle, WA, August 1999.

[24] D. Estrin, D. Culler, K. Pister, G. Sukhatme, Instrumenting the physical world with pervasive networks, Pervasive Computing 1(1) (2002) 59-69.

[25] J. Feder, Random sequential adsorption, Journal of Theoretical Biology 87(2) (1980) 237-254.

[26] D. Ganesan, R. Govindan, S. Shenker, D. Estrin, Highly resilient, energy efficient multi-path routing in wireless sensor network, In Mobile Computing and Communications Review(MC2R) 1(2) (2002).

[27] J. J. Garcia-Luna-Aceves, Loop-free routing using diffusing computations, IEEE Transactions on Networking 1(1) (1993).

[28] S. Ghiasi, A. Srivastava, X. Yang, M. Sarrafzadeh, Optimal energy-aware clustering in sensor networks, Sensors 2 (2002) 258-269.

[29] L. Girod, N. Bulusu, D. Estrin, Scalable coordination for wireless sensor networks: Self-configuring localization systems, Proceedings of the International Symposium on Communication Theory and Applications(ISCTA), July 2001.

[30] M. Gerla, J. T. Tsai, Multicast, mobile, multimedia radio network, ACM-Baltzer J. Wireless Networks 1(3) (1995) 255-265.

[31] P. Gupta, P. Kumar, The capacity of wireless networks, IEEE Transactions on Information Theory, IT-46(3) (2000) 388-404.

[32] J. A. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, B. Heile, IEEE 802.15.4: A developing standard for low-power, low-cost wireless personal area networks, IEEE Network 15(5) (2001) 12-19.

[33] J. C. Haartsen, The bluetooth radio system, IEEE Personal Communications Magazine, Feburary 2000, 28-36.

[34] Z. J. Haas, M. R. Pearlman, The performance of query control schemes for the zone routing protocol, ACM/IEEE Transactions on Networking 9(4), August 2001, 427-438.

[35] C. Hedrich, Routing information protocol, Request for Comments(RFC) 1058, June 1988.

[36] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Covindan, D. Estrin, D. Ganesan, Building efficient wireless sensor networks with low-level naming, Proceedings of the 20th ACM Symposium on Operating Systems Principles(SOSP), October 2001, 146-159.

[37] W. R. Heinzelman, J. Kulik, H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks, Proceedings Of the ACM Mobile Computing and Networking(MOBICOM'99), Seattle, WA, August 1999, 174-185.

[38] W. R. Heinzelman, A. Chandrakasa, H. Balakrishnan, Energy-efficient communication protocols for wireless micro-sensor networks, Proceedings of the Hawaiian International Conference on System Sciences(HICCS'33), Hawaii, January 2000.

[39] L. Hester, Y. Huang, O. Andric, A. Allen, P. Chen, neuRFon Netform: a self-organizing wireless sensor network, http://citeseer.nj.nec.com/572360.html

[40] J. Hill, M. Horton, R. Kling, L. Krishnamurthy, The platforms enabling wireless sensor networks, Communications of the ACM 47(6) (2004) 41–46.

[41] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, Proceedings Of the ACM Mobile Computing and Networking(MOBICOM'00), boston, MA, August 2000.

[42] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for WSNing, IEEE/ACM Transactions on Networking 100 (1) (2003).

[43] Intel Research, New Computing Frontiers – the Wireless Vineyard, http://www.intel.com/technology/techresearch/research/rs01031.htm

[44] K. H. Jones, K. N. Lodding, S. Olariu, L. Wilson, C Xin: Biology-inspired distributed consensus in massively-deployed sensor networks. Proceedings of the 4th International Conference on Ad-hoc Network and Wireless, October 2005, 99-112.

[45] J. M. Kahn, R. H. Katz, K. S. J. Pister, Mobile networking for Smart Dust, Porceedings of the ACM International Conference on Mobile Computing and Networking(MOBICOM'99), Seattle, WA, August 1999.

[46] P. Karn, MACA a new channel access method for packet radio, In ARRL/CRRL Amateur Radio 9th Computer Networking Conference, September 1990, 134-140.

[47] O. Kasten, Wireless Communication Energy Consumption, http://www.inf.ethz.ch/kasten/research/bathtub

[48] L. Kleinrock, J.A. Silvester, Optimum transmission radii for packet radio networks or why six is a magic number, In IEEE National Telecommunication Conference, December 1978.

[49] J. Kulik, W. Heinzelman, H. Balakrishnan, Negotiation-based protocols for disseminating information in wireless sensor networks, Wireless Networks, March 2002.

[50] M. Kumar, L. Schwiebert, M. Brockmeyer, Efficient data aggregation middleware for WSNs, The 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems, Fort Lauderdale, FL, October 2004.

[51] P. Jacquet, P. Muhlethaler, A. Qayyum, Optimized link state routing protocol, IETF MANET working group, Internet Draft, November 1998.

[52] J. M. Jaffe, F. M. Moss, A responsive routing algorithm for computer networks, IEEE Transactions on Communications, COM-30(7), July 1982.

[53] D. B. Johnson, D. A. Maltz, Dynamic source routing in ad hoc wireless networking, In Mobile Computing, Kluwer Academic Press, 1996, 153-181.

[54] K. Langendoen, N. Reijers, Distributed localization algorithm, Embedded Systems handbook, CRC Press, Boca Raton, FL, 2005.

[55] N. Li, J. C. Hou, L. Sha, Design and analysis of an MST-based topology control algorithm, Proceedings of the 22nd International Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM'03), San Francisco, CA, April 2003.

[56] M. Lin, K. Marzullo, S. Masini, Gossip versus deterministic flooding: Low message overhead and high reliability for broadcasting on small networks, UCSD Technical Report RT CS99-0637, http://citeseer.nj.nec.com

[57] S. Lindsey, C. Raghavendra, K. Sivalingam, Data gathering algorithms in sensor networks using energy metrics, IEEE Transactions on Parallel and Distributed Systems 13(9) (2002) 924-935.

[58] J. Lipman, P. Boustead, J. Chicharo, Reliable minimum spanning tree flooding in ad hoc networks, Proceedings of the IEEE Vehicular Technology Conference(VTC 2005), September 2005.

[59] K. Lougheed, Y. Rekhter, Border gateway protocol 3 (BGP-3), Request for Comments(RFC) 1267, October 1991.

[60] LAN MAN Standards Committee of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical Layer (PHY) specification, IEEE Std 802.11-1997 edition, 1997.

[61] H. O. Marcy, J. R. Agre, C. Chien, L. P. Clare, N. Romanov, A. Twarowski, Wireless sensor networks for area monitoring and integrated vehicle health management applications, Rockwell Science Center, Thousand Oaks, CA, Technical Report AIAA-99-4557, 1999.

[62] K. Martinez, J. K. Hart, R. Ong, Environmental sensor networks, IEEE Computer 37(8) (2004) 50-56.

[63] S. Meguerdichian, F. Koushanfar, G. Qu, M. Potkonjak, Exposure in wireless ad hoc sensor networks. Proceedings of the 7th Annual International Conference on Mobile Computing and Networking(MOBICOM'01), July 2001.

[64] S. Meguerdichian, F. Kowshanfar, M. Potkonjak, M. Srivastava, Coverage problems in wireless ad hoc sensor networks. The 20th Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM'01), April 2001, 1380-1387.

[65] P.M. Merlin, A. Segall, A failsafe distributed routing protocol. IEEE Transactions on Communications, COM-27(9), September 1979, 1280-1288.

[66] http://www.stanford.edu/class/ee321/ho

[67] http://www.darpa.mil/mto/mems

[68] S. Murthy, J. J. Garcia-Luna-Aceves, An efficient routing protocol for wireless networks, MONET 1, October 1996, 183-197.

[69] R. Negpal, D. Coore, An algorithm for group formation in an amorphous computer, AI Memo 1626, MIT, 1997.

[70] J. Ni, S. Chandler, Connectivity properties of a random radio network, Proceedings of the IEE Communications 141, August 1994, 289-296.

[71] S. Y. Ni, Y. C. Tseng, Y. S. Chen, J. P. Sheu, The broadcast storm problem in a mobile ad hoc network, Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking(MOBICOM'99), Seattle, WA, August 1999, 152-162.

[72] D. Niculescu, Positioning in ad hoc sensor networks, IEEE Network 18(4) (2004) 24-29.

[73] National Research Council, Embedded, everywhere: a research agenda for systems of embedded computers, http://www.nap.edu/catalog/10193.html

[74] G. Y. Onoda, E. G. Liniger, Experimental determination of the random parking limit in two dimensions, The American Physical Society, January 1986.

[75] S. Olariu, A. Wadaa, L. Wilson, M. Eltoweissy, Wireless sensor networks: leveraging the virtual infrastructure, IEEE Network 18(4) (2004) 51-56.

[76] V. D. Park, M. S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM'97), Kobe, Japan, April 1997, 1405-1413.

[77] S. Park, I. Locher, A. Savvides, M. B. Srivastava, A. Chen, R. Muntz, S. Yue, Design of a wearable sensor badge for smart kindergarten, Proceedings of the 6th International Symposium on Wearable Computers, Seattle, WA, October 2002.

[78] http://may.cs.ucla.edu/projects/parsec

[79] G. Pei, M. Gerla, T, -W. Chen, Fisheye state routing: a routing scheme for ad hoc wireless networks, Proceedings of the 4th International Workshop on Implicit Computational Complexity(ICC'02), New York, NY, April 2002.

[80] C. E. Perkins, E. M. Royer, Ad hoc on-demand distance vector routing, Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications(WMCSA'99) 3, New Orleans, LO, February 1999, 90-100.

[81] T. K. Philips, S. S Panwar, A. N. Tantawi, Connectivity properties of a packet radio network model, IEEE Transactions on Information Theory 35, September 1989, 1044-1047.

[82] J. Polastre, R. Szewcyk, A. Mainwaring, D. Culler, J. Anderson, Analysis of wireless sensor networks for habitat monitoring, in Wireless Sensor Networks, Kluwer Academic Press, 2004, 399-423.

[83] Y. Pomeau, Some asymptotic estimates in the random parking problem, Journal of Physics A: Mathematical and General 13 (1980) 193-196.

[84] G. J. Pottie, W. J. Kaiser, Wireless integrated network sensors, Communications of the ACM 43(5) (2000) 51-58.

[85] A. Renyi, On a one-dimensional problem concerning random space-filling. Publications of Mathematics Institute, Hungrian Academy of Sciences 3 (1958) 109-127.

[86] http://www.networks.digital.com

[87] K. Ryokai, J. Cassell, StoryMat: A play space for collaborative storytelling, ACM Conference on Human Factors in Computing Systems(CHI'99), October 1999.

[88] L. Schwiebert, S. D. S. Gupta, J. Weinmann, Research challenges in wireless networks of biomedical sensors, Proceedings of the 7th Annual International Conference on Mobile Computing and Networking(MOBICOM'01), July 2001, 151-165.

[89] R. C. Shah, J. Rabaey, Energy aware routing for low energy ad hoc sensor networks, Proceedings of the IEEE Wireless Communications and Networking Conference(WCNC'02), Orlando, FL, March 2002.

[90] P. Saffo, Sensors, the next wave of innovation, Communications of the ACM, 40(2) (1997) 93-97.

[91] C.-C. Shen, C. Srisathapornphat, C. Jaikaeo, Sensor information networking architecture and applications, IEEE Personal Communications, August 2001, 52-59.

[92] G. T. Sibley, M. H. Rahimi, G. S. Sukhatme, Robomote: A tiny mobile robot platform for large-scale ad hoc sensor networks, Proceedings of the IEEE International Conference on Robotics and Automation(ICRA'02), May 2002.

[93] M. Sichitiu, C. Veerarithiphan, Simple accurate synchronization for wireless sensor networks, Proceedings of the IEEE Wireless Communications and Networking Conference(WCNC'03), March 2003.

[94] S. Singh, C. S. Raghvavendra, PAMAS: power aware multi-access protocol with signaling for ad hoc networks, ACM Computer Communication Review, 28(3), July 1998, 5-26.

[95] F. Sivrukaya, B. Yener, Time synchronization in sensor networks: a survey, IEEE Network 18(4) (2004) 45-50.

[96] K. Sohrabi, G. Pottie, Performance of a novel self-organization protocol for wireless ad-hoc sensor networks, Proceedings of the IEEE Vehicular Technology Conference(VTC'99), September 1999.

[97] K. Sohrabi, G.Pottie, A self organizing wireless sensor network, Proceedings of the 39th Annual Allerton Conference Communicaton, Control, and Computing, Urbana, IL, October 1999.

[98] K. Sohrabi, J. Gao, V. Ailawadhi, G. Pottie, Protocols for self-organization of a wireless sensor network, IEEE Personal Communications, October 2000, 16-27.

[99] M. Srivastava, R. Muntz, M. Potkonjak, Smart Kindergarten: Sensor-based wireless networks for smart developmental problem-solving environments, Proceedings of the 7th Annual International Conference on Mobile Computing and Networking(MOBICOM'01), July 2001.

[100] M. Stemm, R. H. Hatz, Measuring and reducing energy consumption of network interfaces in hand-held devices, IEICE Transactions on Communications, E80-B(8), August 1997, 1125-1131.

[101] R. Szewczyk, E. Osterweil, J. Polatre, M. Hamilton, A. Mainwaring, D. Estrin, Habitat monitoring with sensor networks, Communications of the ACM, 47(6) (2004) 34-40.

[102] R. Szewczyk, J. Polastre, A. Mainwaring, J. Anderson, D. Culler, An analysis of a large scale habitat monitoring application, Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems, November 2004.

[103] K. Takagi, L. Kleinrock, Optimal transmission ranges for randomly distributed packet radio networks, IEEE Transactions on Communination(COM-32), March 1984, 246-257.

[104] A. S. Tanenbaum, Computer Networks, Prentice Hall Press, 1998.

[105] S. Tilak, N. B. Abu-Ghazaleh, W. Heinzelman, A taxonomy of wireless micro-sensor network models, Mobile computing and Communications Review 6(2) (2004) 28–36.

[106] P. F. Tsuchiya, The landmark hierarchy: A new hierarchy for routing in very large networks, ACM Computer Communication Review 18 (1988) 35-42.

[107] C. Ulmer, organization techniques for wireless in-situ, sensor networks, http://www.craigulmer.com/research

[108] A. Wadaa, S. Olariu, L. Wilson, K. Jones, Q. Xu, On training wireless sensor networks, Proceedings of the 3rd International Workshop on Wireless, Mobile and Ad Hoc Networks(WMAN'03), Nice, France, April 2003.

[109] A. Wadaa, S. Olariu, L. Wilson, K. Jones, M. Eltoweissy, Training a sensor networks, MONET, January 2005.

[110] L. Wang, S. Olariu, Cluster maintenance in mobile ad hoc networks, Wireless Communications and Mobile Computing, September, 2004.

[111] B. Warneke, M. Last, B. Leibowitz, K. Pister, Smart Dust: communicating with a cubic-millimeter computer, IEEE Computer 34(1) (2001) 44-51.

[112] http://www.xbow.com

[113] Q. Xu, R. Ishak, S. Olariu, S. Salleh, On asynchronous training in wireless sensor networks, Proceedings of the 3rd International Conference on advances in mobile Multimedia, September 2005.

[114] F. Xue, P. R. Kumar, The number of neighbors needed for connectivity of wireless networks, Wireless Networks, April 2002.

[115] X. Yang, K. G. Ong, W. R. Dreschel, K. Zeng, C. S. Mungle, C. A. Grimes, Design of a wireless sensor network for long-term in-situ monitoring of an aqueous environment, Sensors 2 (2002) 455-472.

[116] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM'02), June 2002.

[117] Y. Yu, R. Govindan, D. Estrin, Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks, UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, May 2001.

[118] Y. Zhang, W. Lee, Intrusion detection in ad hoc networks, Proceedings of the 6th Annual International Conference on Mobile Computing and Networking(MOBICOM'00), August 2000.

[119] W. Zhang, G. Cao, Optimizing tree reconfiguration to track mobile targets in sensor networks, Proceedings of the 4th International Symposium on Mobile Ad Hoc Networking and Computing(MOBIHOC'03), June 2003.

[120] V. V. Zhirnov, D. J. C. Herr, New frontiers: self-assembly and nano-electronics, IEEE Computer 34(1) (2001) 34-43.

[121] S. Zou, I. Nikolaidis, J.J. Harms, Efficient data collection trees in sensor networks with redundancy removal, Proceedings of the 4th Scandinavian Workshop on Wireless Ad-hoc Network(ADHOC'04), May 2004, 252-265.

VITA

Qingwen Xu was born in Beijing, China on September 19, 1969. He received his Bachelor of Science in Economics from WuHan University, China, in July 1991. He worked as an assistant auditor in Beijing Auditing Bureau from August 1991 to November 1994 and as a stockbroker in ZhongHaiHeng Inc. from December 1994 to July 1997. He received his Master of Science in Computer Science from Wake Forest University in June 1999. In September 1999, he started working on his Ph.D Degree in Computer Science at Old Dominion University, Norfolk Virginia 23529.

Typeset using LaTeX.