DESIGN AND IMPLEMENTATION OF LARGE-SCALE WIRELESS SENSOR NETWORKS FOR ENVIRONMENTAL MONITORING APPLICATIONS

Jue Yang

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

May 2010

APPROVED:

Xinrong Li, Major Professor
Yan Huang, Co-Major Professor
Miguel F. Acevedo, Committee Member
Shengli Fu, Committee Member
Kamesh Namuduri, Committee Member
Ian Parberry, Chair of the Department of Computer Science and Engineering
Costas Tsatsoulis, Dean of the College of Engineering
Michael Monticino, Dean of the Robert B. Toulouse Graduate School

Yang, Jue. Design and Implementation of Large-Scale Wireless Sensor Networks for Environmental Monitoring Applications. Doctor of Philosophy (Computer Science), May 2010, 100 pp., 2 tables, 36 illustrations, bibliography, 79 titles.

Environmental monitoring represents a major application domain for wireless sensor networks (WSN). However, despite significant advances in recent years, there are still many challenging issues to be addressed to exploit the full potential of the emerging WSN technology. In this dissertation, we introduce the design and implementation of low-power wireless sensor networks for long-term, autonomous, and near-real-time environmental monitoring applications. We have developed an out-of-box solution consisting of a suite of software, protocols and algorithms to provide reliable data collection with extremely low power consumption. Two wireless sensor networks based on the proposed solution have been deployed in remote field stations to monitor soil moisture along with other environmental parameters. As parts of the ever-growing environmental monitoring cyberinfrastructure, these networks have been integrated into the Texas Environmental Observatory system for long-term operation. Environmental measurement and network performance results are presented to demonstrate the capability, reliability and energy-efficiency of the network.

ACKNOWLEDGMENTS

First of all I would like to deliver my sincere gratitude to my advisor, Dr. Xinrong Li for his full support and supervision throughout my five years doctoral study. I am grateful for the tremendous time and energy he spent in steering my research. Indeed, without his support, I would never have had the chance to conduct research on the promising field of wireless sensor network, which may become my lifetime career.

I would also like to thank my committee members, Dr. Yan Huang, Dr. Miguel F. Acevedo, Dr. Shengli Fu and Dr. Kamesh Namuduri for their invaluable suggestions and advice during the entire process of research and project.

I would say many thanks to all my former and current colleagues in the CRI/TEO project group. This research is a team effort. I would not be able to accomplish this project and research without the help from all the great team members. Special thanks is given to Chengyang Zhang for his fantastic website which allows me to manage and analyze data with little effort, and to Sanjaya Gurung for his great work in process of system deployment and maintenance.

I also want to express my gratitude to all Department of Electrical Engineering faculty and staff with whom I have a good fortune to interact, particularly to Chair of the Department, Dr. Murali Varanasi who has created a favorable environment for research.

I dedicate this dissertation to my family, who have always been there for me. I feel a deep sense of gratitude for my parents who have always given me the love and support. This work is also in memory of my grandpa, a true warrior who has encouraged me to succeed in achieving high goals.

In case I forgot to mention the names of those people who helped me directly or indirectly in this work, I would like to thank all of them also.

CONTENTS

CHAPTER 1

INTRODUCTION

The rapid advances in electromechanical (MEMS) sensor technology, growing popularity of wireless networks and continuing development of embedded computing devices have lead to the emergence of wireless sensor networks (WSN). It has raised considerable interest in the research community as it has great potential to revolutionize many science and engineering domains.

A wireless sensor node, also known as a "mote" is a low-cost, battery-powered embedded computing device containing a short-range radio transceiver, a processing unit or microcontroller and certain types of analog and digital interfaces, to which a variety of sensing units such as thermistors, photoresistor and accelerometers can be adapted. Through onboard wireless transceiver, the sensor nodes can communicate with each other and automatically organize themselves into an ad-hoc network, which does not rely on a preexisting infrastructure, such as base transceiver stations (BTS) in cellular networks or access points (AP) in wireless local area networks (Wi-Fi). Instead, the network is established and maintained dynamically based on the deployment environment, radio connectivity and states of individual nodes. This ability of self-configuration makes WSN easy to install, expand and maintain, as well as resilient to the failure of individual nodes. The resulting network is orders of magnitude more expansive than current computer networks and will extend the existing Internet deep into the physical environment and make Internet of Things (IOT) possible.

In recent years, WSNs are being developed for a wide variety of applications including environmental monitoring [27, 28, 29, 30], home automation [65], remote healthcare and patient monitoring [63], industrial sensing and diagnostics [67, 69], asset tracking and supply chain management [64]. In this research, I pay particular attention to the environmental

monitoring applications as they represent a major class of WSN applications with enormous potential benefits for the whole society and may become the killer application for wireless sensor networks. In what follows, this dissertation explores the advantages of applying WSN technologies to environmental monitoring systems and then presents the state of the art of such systems.

1.1. Wireless Sensor Networks for Environmental Monitoring

Sensor networks have been widely envisioned to being able to provide long-term, near-real-time observations at unprecedented fine spatiotemporal resolution, which makes it possible for environmental scientists to measure properties that have not been observable previously. The idea of automating the collection of physical data in order to monitor environments is not new; ecological and environmental scientists have been developing and utilizing environmental observatories, consisting of a variety of sensors, sophisticated computational resources and informatics, to observe, model, predict, and ultimately help preserve the health of the natural environment. Instrumenting the physical world at scale with pervasive networks of embedded sensors becomes more important as we recognize that the natural world is inextricably linked to the human society to form an extremely complex ecosystem.

Traditionally, researchers make regular field trips to collect samples of soil, water, and plants etc. and analyze samples in laboratory with sophisticated instruments. Such a technique is still widely used today for the analysis of chemical and biological contaminants, and as a reference technique to assess the performance of in-situ sensors. With the growing number of inexpensive, portable and reliable sensors and loggers available in the market, scientists are also capable of examining samples and investigating environment in-situ. However, this labor intensive method is unable to assess variables with fast temporal changes and may even miss readings due to inaccessibility during extreme physical and weather conditions.

More recently, weather stations with data-logging capability have been developed. Sensors are connected to a data logger with the ability to store data that can be retrieved at a later stage. Though manual download is still necessary, weather stations provide continuous

monitoring for a wealth of physical phenomena with fine-grained time resolution. To improve the responsiveness of the system, telemetry system using cellular networks such as global system for mobile communications (GSM) is widely used together with weather stations as mobile network coverage has been continually enlarged and broadened.

The main disadvantage with the use of weather stations is the issue of cost. In particular, installation, maintenance and communication costs are prohibiting such systems from being deployed at scale. For monitoring parameters such as air temperature and humidity which evolve rapidly over time but usually slowly over space, taking samples at weather station would be sufficient; however, for parameters exhibiting high space variability such as soil moisture, continuous monitoring at several locations is necessary. Additionally, weather stations are usually equipped with energy hungry devices such as cellular modem. When line power is not obtainable, they must be relying on large capacity batteries and solar panels, and installed at spot with adequate sunlight. Sometimes surrounding trees have to be cut down in order to provide an open field for the deployment of a weather station. The selection of deployment location often results biased measurements, for example, inflated air temperature and understated soil moisture data.

Wireless sensor networks, on the other hand, allow for the collection of data at a high spatial and temporal resolution at low cost. Although they remain expensive at the moment as they have not been manufactured in high volume, they are at least one order of magnitudes cheaper than traditional weather stations connected to cellular networks. Wireless sensors are also designed to be energy-efficient; a sensor node can operate for months or even years with two AA batteries. The resulting small form factor presents minimal intrusiveness to the environment. Combining with their ability of self-organizing, WSN presents incredible flexibility and can be deployed directly on the area of interests. A sensor's intimate connection with its immediate physical environment allows each sensor to provide localized measurements and detailed information that is hard to obtain through traditional instrumentation.

Of course, due to scarcity of energy, not all sensors can be excited and supported by a WSN node. Complex signal processing algorithm may not be implementable due to the lack of computational power and memory space. Moreover, the network formed by wireless sensors is only a local area network. It still requires other types of networks to gain internet access. Consequently, the best solution to environmental monitoring would be achieved by the integration of wireless sensor network, which offers observations with fine grained spatial resolution, and weather station which provides a more complete set of measurements as well as telecommunication for remote access.

1.2. Related Works

A number of WSNs have been deployed for monitoring a diversity of environmental physics [28]. In one of the pioneering project, a wireless sensor network was used to monitor the microclimates in and around the nesting burrows of seabirds on Great Duck Island (GDI) in Maine [27]. Wireless motes with an array of sensors to measure, among other things, temperature, light levels, humidity, and infrared radiation were placed in the burrows before the nesting season begins. Since virtually no maintenance was needed afterwards, the impact on the wildlife being monitored would be minimal. The non-intrusive property of WSNs is of particular benefit in habitat studies where any human presence is likely to be disruptive or where a species is particularly sensitive to humans.

The habitat monitoring system utilized a tiered architecture. Motes originating measurement samples lied at the lowest level. These sensor nodes, powered by only two AA batteries, performed sensing and transmitting tasks on the basis of duty cycle. Within each patch, a single-hop, star-topology network is organized. In such a network, motes were simply reporting data to a gateway during scheduled communication periods. Data is only communicated in one direction and there is no dependency on surrounding motes for relaying packets in a multihop manner. The gateways were responsible for gathering data from their patches and then forwarding data through a local transit network to the remote base station (BS) which provided Internet connectivity and data logging. Thirty-two motes were

deployed for four weeks as of the writing of the paper and they had a calculated life time of six months based on the analysis of energy budget and power consumption.

Another example is the Volcan Tungurahua project [34] which used a WSN to monitor volcanic activity by specially-constructed microphones to monitor low-frequency acoustic signals emanating from the volcanic vent during eruptions. Volcanic measurements are often sampled continuously at rates of 40 Hz or more, far greater than the low frequencies used in other environmental monitoring studies. In the project, three Mica2 motes [59] equipped with infrasonic microphone nodes had been recording signals at 102Hz for over 54 hours. Data was transmitted to an aggregation node, another Mica2 mote which relayed the data over a 9 km wireless link to a laptop at the volcano observatory via a long-haul wireless modem. To establish a common time base for cross-correlating the signals captured by each infrasound node, a dedicated global positioning system (GPS) node was used to broadcast time synchronization messages. The small-scale short-term deployment demonstrated the feasibility of using wireless sensors for volcanic studies; however, a number of challenges must be tackled for expanding the network size and extending periods of running time. Careful power management techniques, such as triggering and in-network event detection, and multiple-hop protocols for time synchronization and data routing must be developed.

There have been several attempts to deploy WSNs in extremely harsh conditions such as glacier environment. The GlacsWeb [27] project aimed at understanding glacier dynamics in response to climate change, in particular, how glaciers contribute to sea-level change by releasing fresh water into the sea. The monitoring system was composed of customized sensor probes embedded in the ice and till, a base station on the ice surface, and reference station 2.5 km away from the glacier with access to electricity. Probes were programmed to wake up every 4 hours and record temperature, strain, pressure, 3-axis orientation and resistivity. The base station was scheduled to collect measurements from probes once a day at a set time. It also logged its location with the differential GPS once a week. All data was then transmitted to the reference station PC by a long range radio modem. A total of

8 probes and a base station were deployed in Briksdalsbreen Glacier, Norway. During the experiment, the base station had experienced power failure as the heavy snow covered the solar panel. A few months after deployment, only three probes out of eight had been able to communicate with the base station. A possible cause of the communication failure was the non ad-hoc design of the system. A multiple hop, self-organizing network would not only ensure scalability but also reduce power consumption.

PermaSense [31] investigated the use of power efficient, multi-hop networking protocols for environmental monitoring. In PermaSense, network topology was established automatically, unlike GlacsWeb which employed predefined topology and communication schedule. To support ultra low power operation, sleeping cycles of the radio receiver and consequently synchronization of the wake periods were implemented. However, despite the great design, the deployment in Swiss Alps was not successfully. The project faced three major challenges including accurate time synchronization in hostile environment, stable and reliable multihop routing and efficient power management.

WSNs can be also employed to provide vital hazard warnings as demonstrated in the Floodnet project [33]. A set of intelligent sensor nodes have been deployed around a stretch of the River Crouch in Essex, UK. The nodes were powered by solar cells in conjunction with batteries and established an ad-hoc 802.11 network based on an adaptive routing algorithm. A special node, the gateway, relayed data back to the base using general packet radio service (GPRS) cellular network. The focus of the research was on the fundamental tradeoff between the need for timely data and the need to conserve energy. The goal is the make the system adaptive so that sampling and reporting rates vary according to different situations. For doing so, adaptive sampling and routing algorithm were developed with an extensive co-design exercise of environmental experts. Based on a flood predictor model, Floodnet effectively prioritized the process of data gathering and prolonged the lifetime of the network. It also demonstrated the potential of using pervasive computer in the environment.

Efforts were also made to exploit WSN technology in soil moisture monitoring as soil has been recognized as the most spatially complex stratum of a terrestrial ecosystem. In [30], a field trial network monitoring surface soil moisture was deployed in Banksia woodland on the sandy Gnangara groundwater mound, north of Perth, Western Australia. The measurements were used for managing the groundwater resource and assessing safe water abstraction levels. The soil moisture sensor network was powered by Mica2 motes with MDA300 sensor boards [59] and Echo20 soil moisture probes [61]. Like most of aforementioned project, soil moisture sensors were organized as a star-topology network centered on a BS node which bridged the WSN and the Internet through a GPRS modem. The communication scheme was enabled by S-MAC [36] which provides low duty-cycle and reliable data deliver to the BS. A novelty of the system is its reactivity to the environment. The soil moisture WSN could react to rain storms: frequent soil moisture readings were collected every 10 minutes during rain but only a few readings were collected during non-raining days. The reactivity was done by including a node with a tipping bucket rain gauge sensor. Rainfall events detected by the node would be disseminated to WSN. This distinctive feature demonstrated how weather station system may interact with WSN and the benefit of such integration as rain gauges have been widely installed in weather stations.

The project Life Under Your Feet adopted a different approach for soil moisture sampling. Deployed at an urban forest, wireless motes measured and then stored soil moisture and temperature in situ every minute. Measurements were saved on each mote's local flash memory and periodically retrieved using a reliable transfer protocol. According to the flash memory available in the motes and amount of observations collected, data could be downloaded weekly or at least bi-weekly without loss. In this sense, the WSN can be regarded as a distributed database. The system also showed the need for sophisticated calibration techniques translating raw sensor measurements to high quality scientific data. The calibrated measurement was published through web services interfaces. Scientists could analyze current and historical data and help manage the sensor network by using analysis tools.

Project	Measurement Type	Power Management	Topology	Size	Duration
Great Duck Island	Habitat Monitoring	Battery , 5.8% duty cycle	single hop	Total 32 nodes	Estimated 6 months
Volcan Tungurahua	Volcano Seismic Monitoring	Battery , always-on	single hop	3	54 hours
GlacsWeb	temperature, movement	Battery , 0.1% duty cycle	single hop	8	a few months
PermaSense	Temperature, water content	Battery, duty cycle	multihop	13	Less than two weeks
SensorScope	Various	Solar panel, 10% duty cycle	multihop	6 - 20	long term
Floodnet	Pressure, water level	Solar panel, always-on	multihop	N/A	N/A
Pinja @ UWA	soil moisture, rainfall	Battery, 1% duty cycle	multihop	7	longest node: 28 days
Life Under Your Feet	temperature, soil moisture	Battery, 1.7% duty cycle	single hop	10	147 days

Fig. 1.1. Summary of environmental monitoring WSNs.

Perhaps the most successful environment monitoring project that had fully implemented and utilized the features of WSNs is the SensorScope [32] project conducted at Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. The goal was to develop an effective out-of-the-box, with minimal requirements regarding network maintenance system for environmental monitoring. An all-purpose sensing station accommodating up to 7 different external sensors was developed, along with a multi-hop data gathering protocol and a synchronized duty-cycling medium access control (MAC) layer that greatly helps in reducing the overall energy consumption. To create a fully autonomous system that could theoretically last forever, SensorScope utilized two layers of rechargeable batteries and solar panel to power the station. 6 outdoor deployments had been carried out ranging in size from 6 to 97 stations, from the EPFL campus to high mountains.

To summarize the above works, I compare their designs based on several key attributes, as shown in Fig. 1.1. The topology in the figure only indicates the internal network organization of a single WSN. The overall system topology involving multiple base stations

or gateways is not considered here. Similarly, the network size only deals with WSN nodes not including base stations. Power management is usually enabled by duty cycling, i.e. sensor alternating between active and sleep states. While techniques such as adaptive sampling could effectively reduce data transmissions, they are unable to bring down energy consumption very much without being combined with duty cycling because they are not able to eliminate idle-listening which is the dominant portion of energy consumption.

1.3. Motivation and Objectives

This research contributes to develop the Texas Environmental Observatory (TEO) [56] which aims to provide near real-time data on environmental conditions through the development of a cyberinfrastructure (CI). In particular, the proposed CI system will expand the existing weather station systems to include WSNs to monitor watershed soil moisture. The new WSN-based soil moisture monitoring system is developed to support long-term hydrologic monitoring and modeling research. Increasing urbanization brings changes to the land cover of a given drainage area, which in turn increases the quantity of water flowing overland and decreases the amount of time to reach peak flow [7], increasing in some cases the risk of flash floods. Hydrologic models are helpful in predicting how changes in land cover in rapidly urbanizing areas translate into changes in the stream flow regime. These models require inputs that are difficult to measure over large areas, especially variables related to storm events, such as soil moisture antecedent conditions and rainfall amount and intensity. In addition, the ability to monitor in real time rapidly changing variables before, during, and after storm events will contribute to the improvement of rainfall estimations from meteorological radar data and enhance hydrological model forecasts. Both increased spatial resolution and real-time monitoring requirements have raised challenges that traditional standalone weather station systems are difficult to overcome. As discussed in Section 1.1, WSNs are ideally suited for such applications by exploiting large-scale deployment of low-cost sensor nodes with flexible structure.

Despite significant advances in recent years, there are still many challenging issues to be addressed to exploit the full potential of the emerging WSN technology. As indicated in Fig. 1.1, most of the deployments are short-term experiments or proof-of-concept demonstrations, instead of long-term autonomous operation to support ongoing work by domain scientists and practitioners. Most of the works utilized infrastructure-based single hop topology consisting of only a few nodes, in contrast to the promise that WSNs can support large-scale deployment with self-organization ability. Among the three multihop environmental monitoring project, PermaSense [31] had not been successfully collecting data over a period of more than two weeks, while the Banksia Soodland project [30] lost seven nodes in the first 16 days of deployment, with only a one single-hop node operating for 28 days. The causes of problems may be unique to each of the projects, but they are all rooted from the same challenges faced by all WSN applications.

A core design challenge in wireless sensor networks is coping with the harsh resource constraints placed on the individual devices. Many constraints derive from the vision that these devices will be deployed in vast quantities and therefore must be small and inexpensive. Embedded processors running at a few MHz with kilobytes of memory must implement complex, distributed, ad-hoc networking protocols. The most difficult resource constraint to meet is power consumption. As physical size decreases, so does energy capacity. The success of SensorScope [32] project mainly benefits from the use of large sensing station with sufficient energy budget. With an average current consumption of 4mA, a sensing station would deplete the 2200mAh battery within one month, if without the support of solar panel. The dependency on complicated energy harvesting modules will inevitably increase the size as well as cost, require more installation effort and above all, limit its usage in areas without enough sunlight.

In addition to the inherent constraints, harsh environmental conditions also raise challenges in designing WSNs for real-world applications. Radio communication reliability, measurement fidelity and crystal clock accuracy are all subject to environment conditions

such as temperature, humidity, and vegetation density. The design should address the system robustness in all conditions so as to allow WSNs to be used in a wide range of application scenarios.

The objective of this research aims at taming the above challenges and providing an out-of-box solution, composed of various software, protocols and algorithms, for long-term, large-scale environmental monitoring applications. The proposed system is required to meet the full definition of a WSN and will be evaluated through the development of TEO project.

1.4. Contributions of the Research

This research strives to tackle the aforementioned challenges and make significant practical contributions in WSN research. The three major contributions of the research are summarized in this section.

First, this research proposes a publicly available cyberinfrastructure which seamless integrates wired and wireless sensors for long-term, remote, and near-real-time monitoring. Most of the deployments described in Section 1.2 are stand-alone WSN-only systems, monitoring very few environmental parameters, instead of being a part of the ever-growing environmental monitoring cyber infrastructure. As a result, it is difficult to consolidate a broad range of sensor data systematically to study the cross-correlation among various environmental parameters. Thus this dissertation demonstrates how to incorporate both wired and wireless sensor into a publicly available environmental monitoring CI that supports computing research and education. Specifically, the datalogger-based weather stations with numerous wired sensors continue to provide a complete set of environmental data while the wireless sensor networks measure one or more variables in depth with fine grained spatiotemporal resolution.

Second, a suite of WSN software is developed, consisting of various networking protocols and associated algorithms that are optimized for environmental monitoring applications. Although many off-the-shelf WSN hardware platforms have been available in the market for many years, considerable amount of software customization and redevelopment

efforts are required to meet application-specific requirements. It is identified in [31] that the major challenging issues in the development of WSN are communication reliability, time synchronization stability, and reduction of power consumption. These issues are addressed throughout the dissertation. In Chapter 3, a stack of networking protocols that provide reliable multihop data delivery is proposed. Energy efficiency is achieved by adopting a hybrid TDMA/CSMA MAC protocol and fine-grained time synchronization. To assure the stability of time synchronization, I conduct a thorough study on estimation algorithms which compensate for clock offset, skew and drift during outdoor deployment. The underlying driver of timer module is also devised to meet the requirements of stability and efficiency. To further reduce energy consumption, especially when network size scales up, I propose a novel error-bounded adaptive sampling algorithm which brings down not only the communication cost but also the sensing overhead.

Last but not least, the design has been empirically evaluated through the deployment of a soil moisture monitoring system. The above-mentioned challenges in WSNs have been widely recognized by the research community, and considerable efforts have been put into the design of protocols in different layers to address them. However, many protocols are only implemented in simulators instead of real-life application environments. Some protocols are evaluated by short-term experiments or proof-of-concept demonstrations, instead of long-term autonomous operation. This dissertation intends to address many practical issues in real-world application scenarios that are often neglected in the existing literature. The experience in dealing with all problems encountered in the process of deployment is shared in this dissertation. The firsthand experience obtained in the process of development and deployment should help other groups in appreciating and anticipating many issues related to real-world WSN applications.

1.5. Dissertation Organization

The rest of the dissertation is organized as follows. Chapter 2 presents the overall framework of the TEO environmental monitoring infrastructure, including the design of

sensor network base station, long-haul telecommunication, and a unified framework for sensor data collection, management, visualization, dissemination, and exchange.

Chapter 3 reveals the design of WSN software and protocol stack. Three major components of the protocols stack, namely medium access control protocol, time synchronization protocol and multihop routing protocol are described.

Chapter 4 discusses clock estimation algorithm for the time synchronization post-processing. A unified formulation of least squares (LS) time synchronization algorithms to estimate clock offset, skew, and drift using both batch and sequential estimators is presented.

The implementation, deployment, field testing results are presented in Chapter 5. At last, the dissertation is concluded with conclusions and a discussion of future work in Chapter 6.

CHAPTER 2

ENVIRONMENTAL MONITORING CYBERINFRASTRUCTURE

Cyberinfrastructure (CI) describes the interconnected systems of advanced data acquisition, data management, data visualization, data dissemination and other computing and information processing services to support computing research and education. This chapter introduces a novel cyberinfrastructure that features (1) soil moisture monitoring with flexible spatial coverage and resolution, (2) seamless integrated wired and wireless sensors, (3) long-term, autonomous, remote, and near-real-time monitoring, (4) publicly available web services for sensor data visualization and dissemination, and (5) remote system monitoring and maintenance. Specifically, this research focuses on the overall framework and the integration of a variety of field devices in the weather stations. The integration is enabled through the introduction of remote field gateway (RFG). The RFG aggregates data from both wired and wireless sensors and provides a uniformed remote data collection service.

2.1. Design Objectives

The new environmental monitoring cyberinfrastructure significantly improves the capability and usability of the traditional observatory system. Some key design objectives are listed in this section.

Soil moisture monitoring with flexible spatial coverage and resolution: In the existing system, all sensors are deployed inside a small fence-enclosed area, a situation typical of many environmental monitoring systems. There is a need to provide flexibility to extend the spatial coverage and adjust the spatial resolution of soil moisture sensors. The spatial coverage of the system is limited by the physical limitation of the length of the cable connecting the sensors to the datalogger. In contrast, the spatial coverage and resolution of wireless sensor network (WSN) can be conveniently configured to be meaningful to domain scientists.

Integration of WSN with existing environmental observatories: Despite their limitations, traditional environmental monitoring systems with various wired sensors are capable of accomplishing many monitoring tasks, and substantial investments are in place to monitor temperature, wind speed and direction, rainfall, and solar radiation. Drastically replacing the existing systems with an immature technology such as the WSN is considered unacceptable to many domain scientists and practitioners. Therefore, it is important to introduce the new WSN technology without disrupting the ongoing operation of environmental observatories through seamless integration of wired and wireless sensors.

Long-term, autonomous, remote, near-real-time environmental monitoring: Many environmental monitoring systems are deployed in remote areas that are inconvenient to access for data retrieval and system deployment and management. Traditionally, a stand-alone field station consists of a datalogger and a variety of sensors. Datalogger is programmed to sample at a fixed rate and data are stored in its internal memory. The data are retrievable via the serial port using a computer. Thus, accessing the data requires a visit to the field station, which is inconvenient and is extremely difficult, if not impossible, during harsh conditions, for example flood events. In addition, it has been recently recognized that many ecological and environmental studies need long-term data collection and management. Thus, environmental monitoring systems need to be survivable in extreme environmental and weather conditions for long-term operation with limited human intervention, making energy harvesting and energy efficiency major design considerations. Near-real-time data collection is another important feature to support time-sensitive environmental studies, which necessitates a convenient yet reliable long-haul wireless communication link.

Publicly available web services for sensor data visualization and dissemination: It is important to make data publicly available to benefit a broad range of entities such as environmental researchers, local citizens and government policy makers, and K–12 teachers and students. In addition, the explosive growth of environmental data collected by a variety

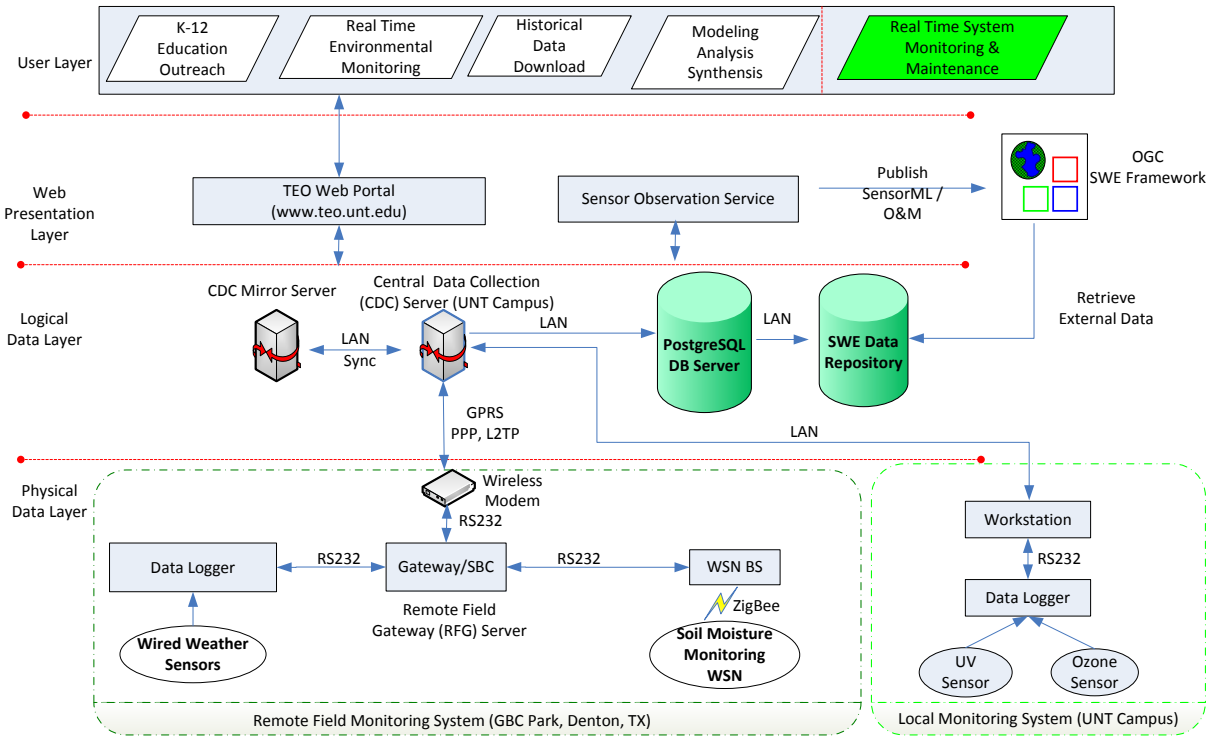

Fig. 2.1. Overall framework of the environmental monitoring cyberinfrastructure.

of sensors in long-term operation necessitates a unified framework for data collection, management, integration, visualization, and dissemination. Such a framework should conform to standards, such as the sensor web enablement (SWE) standard proposed by the open geographic consortium (OGC) [8], to enable data exchangeability and interoperability.

Remote system status monitoring and management: For environmental monitoring systems deployed in remote areas, remote monitoring of system status is extremely useful for system development, debugging, and maintenance purposes. Thus, various system status data need to be carefully defined and collected together with the environmental sensor data. Furthermore, it is important to remotely adjust system configurations and update and upgrade software programs.

2.2. Overall Architecture

The new environmental monitoring cyberinfrastructure can be divided into four major layers as shown in Fig. 2.1, including physical data layer, logical data layer, web presentation

layer, and user layer. Such a layered approach makes it possible to implement the system in a flexible, extensible, and efficient way. At the physical data layer, a variety of sensors are used to monitor environmental parameters. Sensor data are transmitted from a monitoring site to a central data collection (CDC) server. To address above objectives, I incorporate a wireless telemetry system, a single-board computer (SBC) as remote field gateway (RFG) server, and a WSN for distributed soil moisture monitoring. The RFG server provides effective control, management, and coordination of two relatively independent sensor systems, i.e., a traditional datalogger-based wired sensor system and the WSN-based wireless sensor system. The Linux-based RFG server supports remote login to allow maximum remote manipulation of the devices in the field such as the SBC, datalogger, and WSN.

At the logical data layer, sensor data collected from the distributed monitoring stations are stored in a postgreSQL database (DB) server. The CDC server acts as an intermediate component to hide the heterogeneity of different physical layer devices and support data validation required by the DB server. The CDC server and its mirror server also archive raw data on local file systems. Daemon programs running on the CDC server pre-process the data before it is inserted into the database, and periodically perform synchronization tasks. An SWE-compliant data repository is installed to enable data exchange, accepting data from both internal DB server and external sources through the OGC web services.

The web presentation layer consists of a web portal, i.e., TEO Online [56], and a sensor web implementation. The web portal serves as a user-friendly interface for data visualization, analysis, syndissertation, modeling, and K–12 educational outreach activities. It also provides useful capabilities for system developers and operators to remotely monitor system status and remotely update software and system configuration, which greatly simplify system debugging and maintenance tasks. I also implement sensor observation services (SOS) at this layer, conforming to the SWE standard to facilitate data exchange. The standard SensorML/O&M data representation makes it easy to integrate sensor data into the existing geographic information systems (GIS) web services and exchange the data with

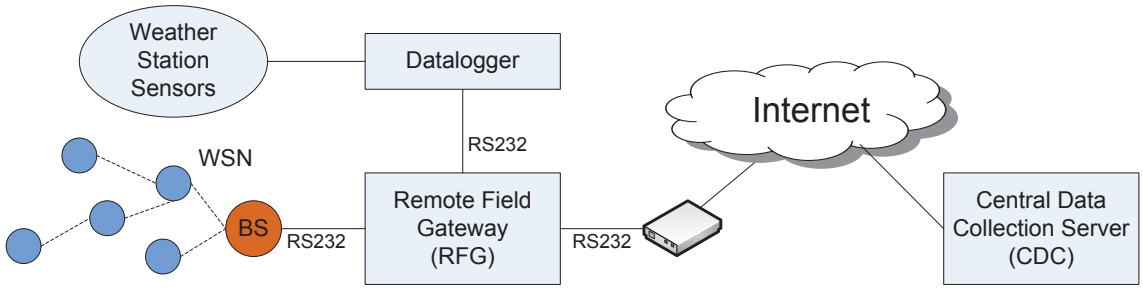

Fig. 2.2. System components of an integrated weather station.

other organizations. The SOS web service will be published to a catalog service in the OGC SWE framework to make it publicly accessible on the Internet.

Finally, the user layer abstracts a variety of needs for education, outreach, research, and system development and management purposes.

2.3. Integration of Wired and Wireless Sensors

As illustrated in Fig. 2.2, current infrastructure on a monitoring site typically includes a datalogger wired with a mixture of environmental sensors, a SBC based RFG, a wireless telemetry device, and a large-scale WSNs to measure near-real-time soil moisture. Each of the system components is described in the next sections.

2.3.1. Datalogger

A datalogger is an electronic programmable instrument or device that records environmental data over time via its built in or external sensors. It is similar to a larger wireless sensor node without the transceiver but much more capable in terms of memory storage, processing power, and interface variety. The following lists the key features of a datalogger.

Large non-volatile memory capacity: Usually a datalogger is programmed to sample at a fixed rate and all measurements are stored in its internal memory. Researchers have to make field trip in order to physically access and download the data. Datalogger is designed to have non-volatile internal memory that could hold up several months' data so that laborious field trips can be less frequent.

High processing power: Some environmental observations with high temporal variance, such as wind speed and wind direction, are only meaningful when data is statistically aggregated. These variables are sampled at relatively high frequency. To reduce memory utilization, signal processing algorithms are employed to calibrate, aggregate, summarize, and compress the data points. Only the processed data abstraction will be stored. Most of modern dataloggers have built-in routines for data processing, which necessitates a powerful central processing unit (CPU).

Support for various interfaces: Dataloggers normally include a wide variety of analog and digital input/output (I/O) ports and sensor excitations, to interface with numerous sensors such as air temperature, air pressure, wind speed and wind direction, rainfall, solar radiation, relative humidity, soil moisture, etc. They are often equipped with a high-resolution analog-to-digital (A/D) conversion with high speed of sample rate. To enable communication with other external devices, they feature a range of peripheral ports such as RS-232, USB and Ethernet port.

However, these qualities do not come without a price. Dataloggers generally consumes more power and are typically powered by a high capacity rechargeable battery, which is recharged by an AC/DC adapter wherever possible, or by a large size solar panel.

2.3.2. Soil Moisture Monitoring WSN

The soil moisture WSN consists of a range of wireless sensor nodes, distributed over the monitoring field, sampling soil moisture periodically. In a single-sink configuration, wireless sensors usually form a data collection tree, rooted at a base station (BS). To establish a functional WSN, the BS broadcasts routing and synchronization beacons regularly. All the other nodes synchronize to the BS node and route data to it. The BS node shares the same type of radio processor board, and is furnished with an RS-232 extension board to interface to other devices. In addition to aggregating data from the WSN to the host it also acts as a portal to monitor network performance and configure network parameters. The detailed discussion of wireless sensor network for soil moisture monitoring is presented in Chapter 3.

2.3.3. Telecommunication

The long-haul wireless communication from the field to the CDC server is currently implemented by using a GPRS modem. GPRS, standing for general packet radio service, is a packet-oriented mobile data service, available to the subscribers of the global system for mobile communications (GSM) cellular networks. The GPRS link is maintained by SBC using the point-to-point (PPP) protocol, a data link protocol commonly used to establish a direct connection between two nodes over serial cable, telephone line, cellular phone, or dial-up networks. Upon boot up, SBC automatically dials to the GPRS network and keeps the link alive during the entire active period. To enable secure system access, the layer 2 tunneling protocol (L2TP) is used to support virtual private network that establishes a secure point-to-point connection between the RFG server and the CDC server through the public Internet. To be energy-efficient, the wireless modem is powered off during the system's sleep period.

As an alternative, a point-to-point Wi-Fi link can be also utilized to connect the RFG and the CDC server. Wi-Fi devices such as Nanostation2 [60], provides license-free communication between a pair of devices with one installed at the remote field site and the other deployed at base site with internet access. With a high power radio frequency (RF) amplifier and high gain directional antenna, they are able to communicate in line of sight (LOS) over tens of miles. The RFG can link to the field site device via local Ethernet connection, and communicate with CDC server through the long range Wi-Fi link. Compared to GPRS network which has data throughput around 15-40 Kbps, Wi-Fi provides much higher data rate at 54 Mbps and more reliable connection. Unlike GPRS, it is also cost effective since there is no monthly service charge. Wherever a LOS connection can be established between the field station and an Internet-accessible site, Wi-Fi communication is preferred to GPRS.

2.3.4. Remote Field Gateway

To seamlessly integrate aforementioned devices in the field, I implement an RFG server using a compact, rugged, ultra-low-power SBC TS-7260 from Technology Systems, Inc. [11]. The SBC provides a standard set of on-board peripherals and includes software power consumption control for on-board peripherals, making it ideal for power sensitive designs, such as solar or battery-powered embedded systems. To minimize energy consumption, the RFG is automatically duty-cycled between the active and power save modes. To provide contingency power support the SBC is installed with an optional battery backup board TS-BAT3, which serves as an embedded uninterruptible power supply (UPS). The devices deployed in the field are commonly equipped with an RS-232 serial port, including data loggers, wireless modem, and the WSN BS node. Thus, with five serial ports onboard, SBC is well suited to serve as a gateway server. Other alternative products in the market typically provide fewer serial ports and have much higher power consumption as compared with TS-7260.

2.3.5. Remote Data Collection Services

The RFG server deployed in the design supports the full-featured Debian GNU/Linux, which may be customized to meet various low-power embedded computing needs. Thus, it is convenient to develop remote data collection services by taking advantage of the software packages that Debian provides, including a complete GNU C/C++ development environment, many Linux services such as PPP, FTP, Telnet, and MySQL database server, and various GNU/Linux libraries and utilities.

The RFG server wakes up periodically to carry out data collection services. Upon boot up, the RFG server executes a series of scripts to initiate various services, including an event logging daemon, a MySQL database server, an FTP server, an SSH terminal, and a Telnet terminal. A PPP daemon is also initiated to establish and maintain a PPP link to the cellular network. The wireless modem is powered on at the same time as the RFG server. Then, several independent data collection processes are started to poll data from the

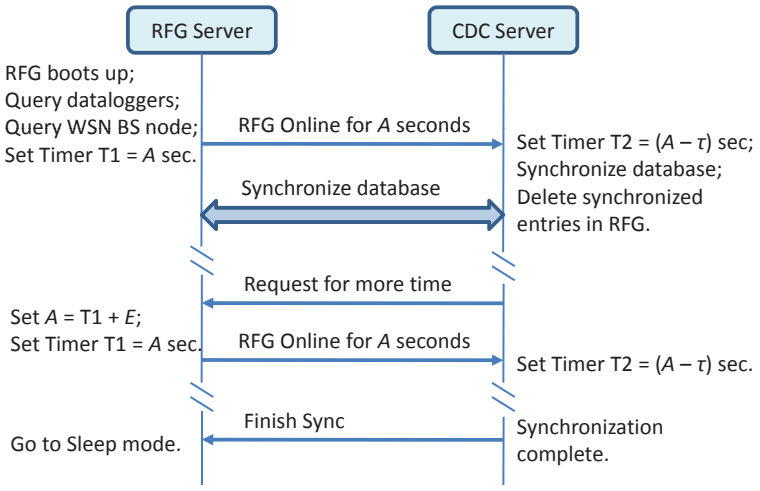

Fig. 2.3. Duty cycle negotiation protocol between the RFG server and the CDC server. The constant $\tau = \text{RTT} + \epsilon$, where RTT is the estimated round-trip time and ϵ is an appropriate guard-band time.

WSN BS node and dataloggers through RS-232 ports. The data collected by the RFG server are inserted into a local MySQL database, instead of being saved in the local file system or directly sent to the CDC server through the wireless modem. The database server provides proficient data management that facilitates efficient data search, enables concurrent data access, minimizes data redundancy, enforces data integrity, and improves data consistency. With the in situ database, sensor data can be readily retrieved through a uniform interface and securely warehoused in the field, even in the event of network failures between the RFG server and the CDC server. After acquiring all data, the RFG server notifies the CDC server that new data is ready for retrieval. The CDC server then synchronizes its database to the RFG database.

The duration of the database synchronization process is random in nature due to the inherent uncertainties in the amount of data to be synchronized and the traffic load condition in the network. Thus, instead of adopting a fixed-length active period, I implement a simple duty cycle negotiation protocol between the RFG server and the CDC server to enhance energy efficiency of the solar-powered remote monitoring system. As shown in Fig. 2.3, the CDC server may request more time when it is needed. If synchronization is finished before

the timer T2 expires, the CDC server sends a Finish Sync command to the RFG server to put it into sleep immediately. The RFG server and the CDC server are protected from potential network failures by the timers T1 and T2, respectively; that is, data collection process is terminated when the timers expire.

2.3.6. Remote Status Monitoring Services

Various system status data are also collected in the same way as sensor data to enable remote monitoring and management of the monitoring systems deployed in the field. Sensor nodes of WSN report system status along with sensor measurement data, such as battery voltage level, network topology data, and network performance statistics. The voltage level of the rechargeable battery, which powers the SBC, wireless modem, and dataloggers inside the station, is monitored by a datalogger and the battery voltage data is reported along with the wired sensor data. The RFG server logs abnormal events in its local file system and reports to the CDC server immediately as long as it is online. Authorized users can adjust system configuration such as duty cycle and sampling rate in near-real-time from the web portal by sending commands to the RFG server through the CDC server. Remote reprogramming of dataloggers and WSN follows the same steps, but because a large amount of data that needs to be downloaded from the CDC server to the RFG server, duty cycling of the RFG server is temporarily disabled. Once the RFG server receives a new program image, sensor nodes are reprogrammed through an over-the-air programming protocol.

2.3.7. Power Management

All the system components in the field are powered by solar energy with a large solar panel and a lead-acid rechargeable battery. The required capacities of the rechargeable battery and the solar panel are determined through power budget analysis. In power budget analysis, average power consumption of each power load device is determined by measuring or estimating the average current draw and the time spent in each of its operating modes. To survive extreme weather conditions in long-term operations, I target at supporting the system with a fully charged battery for at least a week without recharging. In the system, the

Table 2.1
Weather Station Power Consumption Analysis

Device	Mode	Current draw (mA)	Duty cycle (%)	Avg. current draw (mA)
SBC	Active	60	8.3	37.8
	Sleep	35	91.7	
Modem	Active	350	8.3	38.2
	Idle	10	91.7	
WSN BS	Active	11	100	11
Datalogger	Active	16	1	53.5
	Idle	38	99	
Total				140.5

battery voltage level is closely monitored as a part of the remote system status monitoring service as described in the previous section. Near-real-time monitoring of such a system status data is important in determining battery efficiency and early detection of severe battery degradation to prevent system failure and the loss of important sensor data.

As discussed in Section 2.1, energy efficiency is one of the major design considerations of remote environmental monitoring systems. Duty cycling provides an effective way to achieve energy efficiency. In the current setup, the RFG server wakes up for 50 seconds every 10 minutes for data collection with a duty cycle of about 8.3%. The wireless modem is powered off during inactive periods. Table 2.1 shows the current draw and duty cycle of the devices deployed inside the GBC station, powered by a solar panel with a peak current of 6.1 A and a 35 Ah lead-acid rechargeable battery. In practice, a lead-acid battery cannot be 100% discharged repeatedly. Therefore, it is necessary to de-rate the battery by some amount, generally 25% [12]. Thus, the battery deployed in the field may support the system for about 7 days without recharging. In general, the capacity of a solar panel should be at least 10 times the average power consumption of the load [12]; the solar panel deployed in the field meets such a requirement.

CHAPTER 3

WSN NETWORKING PROTOCOLS

The strength of wireless sensor networks (WSN) lies in their flexible network structure, made possible by a variety of networking protocols in different layers. Designing efficient and reliable communication protocols for wireless sensor networks in outdoor monitoring applications is a challenging task, due to various constraints on the sensor platform and uncertainty and dynamics of the environment. This chapter begins with analyzing design requirements of protocols for environmental monitoring applications. Then three major components of the proposed protocols stack are described, namely medium access control (MAC) protocol, time synchronization protocol, and multihop routing protocol. Though still presented in a layered fashion, the protocols are tightly coupled and cross-layer interaction is pervasive in this design.

3.1. Design Requirements of Environmental Monitoring Sensor Networks

Generally speaking, the purpose of environmental monitoring sensor networks is to collect authentic measurements of varying environmental conditions over a long period of time. The following summarizes a set of core design requirements for environmental monitoring sensor networks.

Long-term energy-efficient operation: Most of ecological studies require measurements on long temporal scales ranging from several months to a few years. Without any energy optimization, a typical battery-powered wireless sensor node can last only for a few days. Alternative energy sources such as solar power may not be easily acquired, especially in forest with dense vegetation. While one can employ batteries with larger capacity, it will inevitably lead to an increased form factor and cost. Therefore, minimizing the power consumption is the ultimate key to sustaining long term operation in challenging conditions.

Reliable data collection: All measurement data should be collected reliably because any data loss may lead to a distorted observation. On the other hand, environmental variables typically exhibit high spatial and temporal correlation and hence application layer protocols may exploit such correlation properties in data compression to reduce the communication load of the network. In such scenarios, reliable data collection is even more important because any packet loss may affect the usefulness of many other packets.

Near-real-time data collection: Environmental parameters typically vary slowly [29, 28] and thus latency can be traded off for energy efficiency in environmental monitoring systems. On the other hand, some monitoring systems are used to provide vital hazard warnings such as flash flood alerts. While real-time constraint is generally hard to satisfy, near-real-time data collection is achievable while keeping energy consumption low. Here the term near-real-time pertains to the delay introduced by network transmission, between the occurrences of consecutive measurements. More specifically, the maximal latency of data delivery should be less than the sensor sampling period.

Scalability: Environmental monitoring applications vary significant in terms of spatiotemporal sampling resolution and scale, depending on the physical phenomenon under observation. Thus, environmental monitoring sensor networks need to be scalable to suit a wide range of application scenarios, especially in large-scale monitoring applications such as watershed soil moisture monitoring.

Fault Tolerance: Environmental monitoring systems are often deployed in remote places that are typically harsh and hard to access. Some sensor nodes may fail or be blocked due to lack of power, physical damage, or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network. Thus, in order to accomplish long-term monitoring in such environments, sensor networks need to be capable of self-configuration and self-organization upon initial deployment as well as self-healing in the events of node failures without frequent user intervention. Furthermore, I also consider the recoverability of the network in case of base station (BS) failure, which is often neglected by

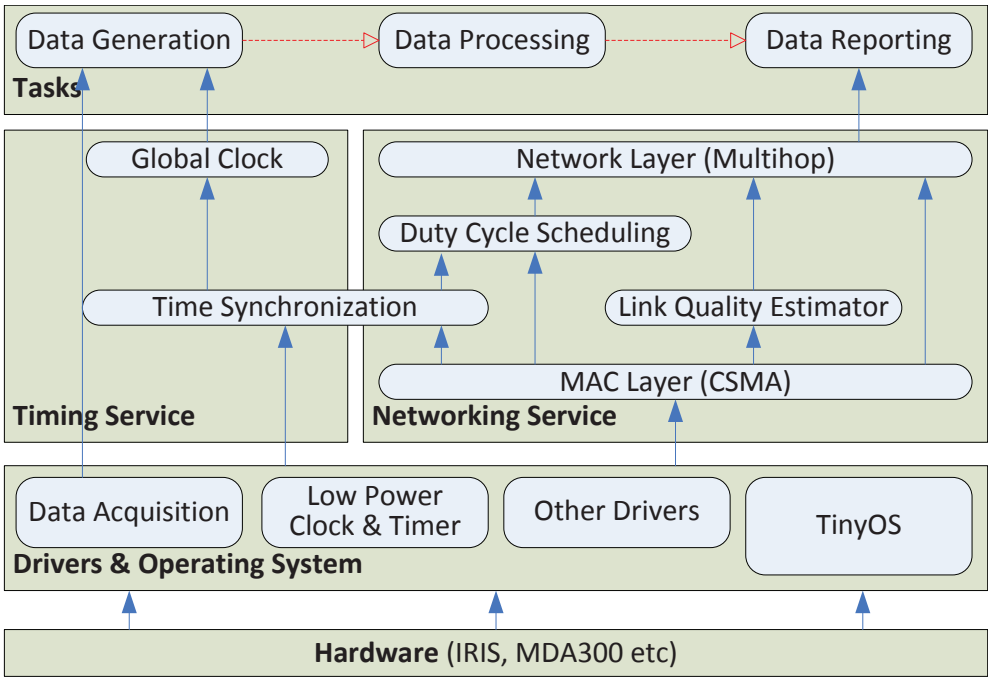

Fig. 3.1. Architecture of WSN software.

the most of the research in the WSN community. In real-life outdoor applications, BS usually faces power shortage, electromagnetic interference (EMI) caused by coexistence devices, or even lightning shock, all of which may cause the BS to reboot or halt. The network should cope with the abnormal behavior of BS and resume normal operation as soon as BS gets back.

Load balance: Most of the environmental monitoring sensor networks have the spanning-tree multi-hop network structure that is rooted in a single sink node. In such networks, sensor nodes closer to the sink are deemed to have higher energy consumption than those nodes that are further away from the sink because of the overhead for relaying messages. Therefore, it is of great importance to balance the load throughout the network to maximize the overall network lifetime.

3.2. Software Architecture

Fig. 3.1 shows the functional block diagram of sensor node implemented in this research. In general, in environmental monitoring applications, every sensor node periodically

carries out three main tasks, including data generation through sensing, data processing, and data reporting through multihop wireless communications. To accomplish the data generation task, sensor readings are collected periodically at certain frequency and sensor data are time-stamped upon sampling, which necessitates global time synchronization in the network. Then, in the data processing task, sensor nodes calibrate, aggregate, summarize, and compress the data. Lastly, during the data reporting task, data are transmitted to the BS or sink node through multihop wireless communications. The data reporting task is enabled by a variety of software services, which implements essential timing, communication, and networking protocols for energy-efficient multihop data collection in distributed networks.

These functional modules are implemented in TinyOS [58], the de-facto operating system for WSN. The TinyOS source tree shipped by the mote manufacturer has included most of the essential drivers for motes and sensor boards. And many networking protocols and services have been implemented in TinyOS in the literature by the WSN community. Although TinyOS promises that designers can easily compose new applications and services by wiring together existing components, implementing a complete working system is not as easy as bringing building blocks together. For example, I tried to replace the default MAC protocol CSMA in TinyOS with the more power-efficient S-MAC [36] and combine it with the default multihop routing protocol. The result turned out to be disappointing: they failed to form a stable routing tree. The default multihop routing protocol uses active probing as well as eavesdropping for link quality estimation, which works well with CSMA. However, S-MAC implements overhearing avoidance mechanism and thus yields inaccurate and erroneous link quality estimation, which is detrimental to proper routing path selection. Moreover, it is necessary to restructure or even redesign most of the building components, especially the networking protocols, to construct a reliably functioning system. It is well understood that in a resource-constrained platform such as wireless sensor node, the stack of protocols should be jointly optimized in order to maximize the overall network performance and to minimize energy consumption. A number of lower-power protocols are available in the

literature, but most of them are not optimized for environmental monitoring applications. In the next three sections, the main components of networking protocol stack as well as other software services and drivers that have been developed in this research for environmental monitoring applications are described in detail.

3.3. Hybrid MAC Protocol for Reliable Data Collection

The MAC layer protocol directly interfaces the radio transceiver hardware through the driver, managing the radio activity, and switching the radio chip among various states. Since the wireless communication is the most energy-consuming operations that a node performs, the design of MAC protocol will determine the overall energy efficiency of the system.

3.3.1. Related Work

In wireless sensor networks, MAC protocols achieve energy efficiency by switching radio transceiver down to a low power mode, for example, standby or sleep mode. In this case, nodes operate on the basis of a duty-cycle, i.e. they alternate between the active and the sleep states with a given pattern. Obviously, a sleep scheduling protocol is needed to coordinate nodes so that neighbors can still communicate together while bring down the energy waste due to idle listening. According to their degree of dependency on synchronization, MAC protocols in WSNs can be categorized as asynchronous, loosely synchronous and fully synchronized protocols [35]. In general, with a greater degree of synchronization between nodes, packet delivery is more energy-efficient due to the minimization of idle listening when there is no communication, better collision avoidance and elimination of overhearing of neighbor conversations.

Asynchronous MAC protocols such as B-MAC [38] uses low power listening (LPL) to reduce idle listening. In LPL, sensor nodes independently follow a sleeping schedule based on target duty cycle and periodically sense channel activity. Before message transmission, the sender is required to transmit a very long preamble to wake up every node in the neighborhood, including the receiver. Since sensor nodes are not synchronized, the preamble must be longer than the sleep period so that the receiver is able to detect it. Although B-MAC

eliminates the overhead of time synchronization, it spends considerable amount of energy in sending the long wakeup preamble. On the other hand, as indicated in Fig. 3.1, time synchronization cannot be removed from the system since it is required by other components as well. Another asynchronous MAC protocol, called Z-MAC [39] employs a distributed slot assignment protocol called DRAND, to ensure unique slot assignment. Prior to normal operation and when topology changes, DRAND protocol needs to be executed to assign slots. The large amount of overhead associated with DRAND makes it unsuitable for networks with frequently changing topology. In addition, Z-MAC is built on top of LPL and B-MAC so that it inherits the shortcomings of B-MAC as discussed earlier.

In synchronized MAC protocols, regardless of their degree of synchronization, the entire time space is divided into slots. Sensor nodes can only communicate during their designated slots and remain silence or even sleep in other slots to conserve energy. Most of the synchronized MAC protocols implement time-slot scheduling algorithms to coordinate slot assignments in the time space and adopt CSMA to coordinate the communication during a particular slot. Loosely synchronous MAC protocols such as S-MAC [36] employ local time synchronization among neighboring nodes to coordinate packet exchanges. Neighboring nodes form virtual clusters and the nodes in the same cluster share a common schedule. The slot is not uniquely assigned but randomly picked. Thus, to transmit a packet, a node competes for the slot not only with the nodes in the same cluster, but also with nodes from other clusters. To avoid possible collision, the RTS-CTS procedure [36] is used in every transmission, which results in a large amount of overhead. For an irrelevant node, which is neither the sender nor the receiver in the cluster, it still needs to wake up in order to receive potential incoming packets. Though S-MAC implements an overhearing avoidance mechanism where a node turns off transceiver when receiving RTS that is not addressed to it, it is inevitable for the node to overhear the RTS packet and waste energy in turning transceiver on and off. T-MAC [37] follows the design of S-MAC but outperforms S-MAC

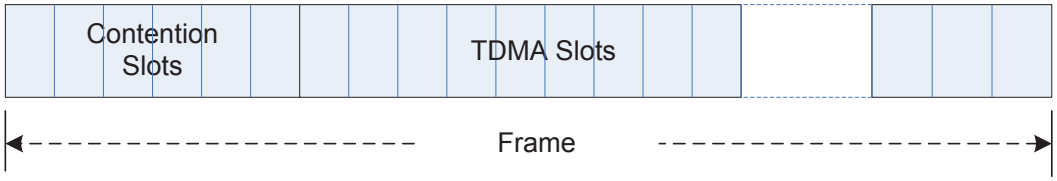

Fig. 3.2. Time slot structure of a frame in the hybrid MAC protocol.

with variable load by introducing adaptive duty cycling scheme. However, it suffers from the same problem as S-MAC.

Fully synchronized protocols, similar to TDMA protocols, are known to provide excellent energy-efficiency due to minimization of idle listening, elimination of overhearing and collision-free operation. In TRAMA [40] and L-MAC [41], the authors assume the availability of global time synchronization considering it an orthogonal problem. Both protocols implement distributed schedule reservation scheme used to establish collision-free operation by negotiating non-overlapping slot across all nodes within 2-hop radius. On the contrary, RT-Link [35] assigns the time slots centrally at the base station and achieve global time synchronization by using out-of-band hardware.

In this research, I develop a fully synchronized MAC protocol that integrates CSMA and duty-cycle scheduling to achieve high energy efficiency to support long-term, low-rate, and large-scale sensor network applications. The hybrid MAC protocol, coupled with the software-based in-band time synchronization protocol, implements a distributed duty-cycle scheduling algorithm to coordinate sensor nodes' sleeping. Compared to the other fully synchronized protocols, this algorithm does not need to guarantee the unique assignment of slot within 2-hop radius in order to achieve collision-free operation. Consequently, this protocol greatly reduces the communication overhead of scheduling protocol. Additionally, it gracefully embeds the slot allocation exchange in the process of upstream parent selection. Hence, the dedicated communication overhead for slot scheduling is virtually zero.

3.3.2. Protocol Description

Similar to most of the synchronized MAC protocols in the literature, the hybrid MAC protocol proposed in this paper equally divides the entire time axis into non-overlapping time frames, each of which is then equally divided into a number of non-overlapping time slots as shown in Fig. 3.2. Two types of time slots are defined in the hybrid MAC protocol, including contention slots and TDMA slots. During contention slots, all nodes in the network are active. They are able to receive packets from neighbors, and contend for the medium to transmit packets. Contention slots are mainly used for broadcasting local information such as neighbors discovery, route selection and time synchronization stamps dissemination. However, unicast packets carrying measurement data can also be transmitted in contention slots with possible packet collision and retransmission. CSMA is the baseline MAC protocol in contention slots to resolve the contention problem.

TDMA slots are intended for delivering upstream unicast data packets. Unicast packets have only one destined node and acknowledgements are often required to ensure the success of delivery. In a spanning tree monitoring network, upstream unicast packets which are transmitted from a sensor node to its parent node represent most of communication traffic since the measurement data must be relayed towards the sink. Hence it is crucial to provide a reliable and collision free environment for such packets. Each TDMA slot can be exclusively owned by a single node in a neighborhood. During TDMA slots, only the owner node is allowed to transmit packets. The parent of the owner is also active to receive and acknowledge packets. Other non-related nodes are normally sleeping with radio transceiver turned off to conserve energy. However, exception occurs when a node would like to overhear these conversations to collect neighbor information or synchronize to the network. Although TDMA slots provide collision-free transmission for unicast packets, CSMA is still used as the underline MAC protocol to avoid any unexpected collision due to inaccurate time synchronization, co-channel interference from other types of devices, etc.

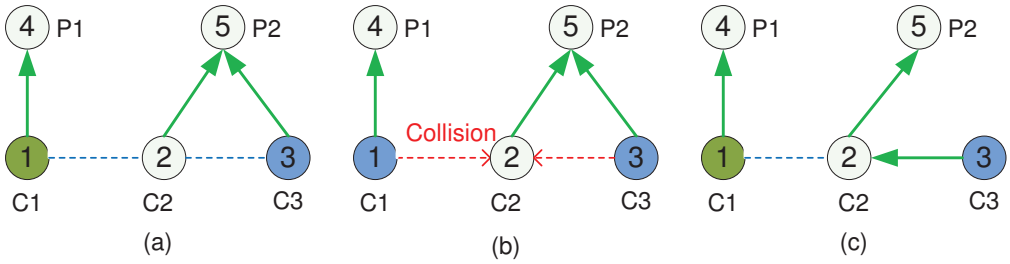

Fig. 3.3. Illustration of collision-free scheduling.

In the current configuration of the hybrid MAC protocol, each frame begins with one contention slot followed by a number of TDMA slots. Initially a node can only transmit in the contention slot before any TDMA slots being assigned to it. After synchronizing to the network and collecting necessary neighborhood information, it will assign a unique TDMA slot to itself according to a distributed slot scheduling protocol detailed in the next section.

3.3.3. Distributed Slot Scheduling Protocol

The distributed slot scheduling protocol (DSSP) aims at assigning a unique TDMA slot to a sensor node for collision-free upstream unicast transmission. Collision-free upstream unicast transmission implies that, during a TDMA slot, the child node is the only sender in its one-hop neighborhood as well as its parent's one-hop neighborhood. Most of the scheduling protocols in the literature, regardless whether they assign slots distributively or centrally, have to guarantee that there will be only one parent-child pair active in any two-hop neighborhood. This is a stricter requirement than the former implication and it is not necessary for collision-free upstream unicast transmission.

Fig. 3.3 shows a partial sensor network consisting of 5 nodes. Dotted line represents connectivity and solid line with arrow represents routing path. The number inside the circle is the assigned TDMA slot number for the node. In this network, C1 and C3 are two-hop neighbors and should have different slot assignments under the requirement of two-hop uniqueness, as shown in Fig. 3.3(a). However, even if C1 and C3 share the same slot, they can still send packets to their respective parents at the same time without worrying about collision, as shown in Fig. 3.3(b). The packets will collide at C2 but it will not affect

anything since C2 is not the receiver and supposed to be sleeping during the slot. However, if C2 is the parent of C3, C1 and C3 must have different slots to avoid collision, as shown in Fig. 3.3(c). Therefore, in order to achieve collision-free upstream unicast, a node must be knowledgeable about the slot allocations of neighbors and neighbors' children. Compared to the two-hop uniqueness which requires the slot allocations of neighbors' all neighbors, this greatly reduces the communication overhead of scheduling protocol. Additionally, the slot allocation exchange can be embedded in the process of upstream parent selection hence the dedicated communication overhead for slot scheduling is virtually zero.

The idea behind the distributed slot scheduling protocol is similar to the RTS-CTS collision avoidance mechanism in the 802.11 standard. In the proposed protocol, each node maintains a network allocation table (NAT), similar to the NAV in 802.11. But instead of keeping the duration field, NAT keeps track of the allocated slot number with its owner ID and a time-to-live (TTL) field, indicating the reservation duration of the slot. As in NAV, the TTL values in NAT will decrement across time. An entry will be removed when its correspondent TTL becomes zero. When a node resets or boots up, it will request for the slot allocations from its neighbors. Upon receiving the request, neighboring nodes will reply back with their slot numbers and their children's slot numbers. This is the only dedicated communication overhead introduced by the scheduling protocol and the one-time overhead will be amortized and approach 0 as time goes by. While passive sniffing can also be used to collect neighborhood information, such an active request method will reduce considerable amount of time for initialization. After parent selection, the child node picks a slot that is not in the NAT, i.e., not occupied by any neighbors and their children and requests for the slot by sending a request (REQ) packet to its parent. The parent node responds with a reply packet (RPL) if the requested slot is not in its NAT, i.e., not in use. Then the child node acknowledges (ACK) the slot allocation and concludes this request. This conversation is conducted during contention slots when all the nodes are awake. Other neighboring nodes eavesdropping the conversation will update their NAT, which in turn is used to ensure that

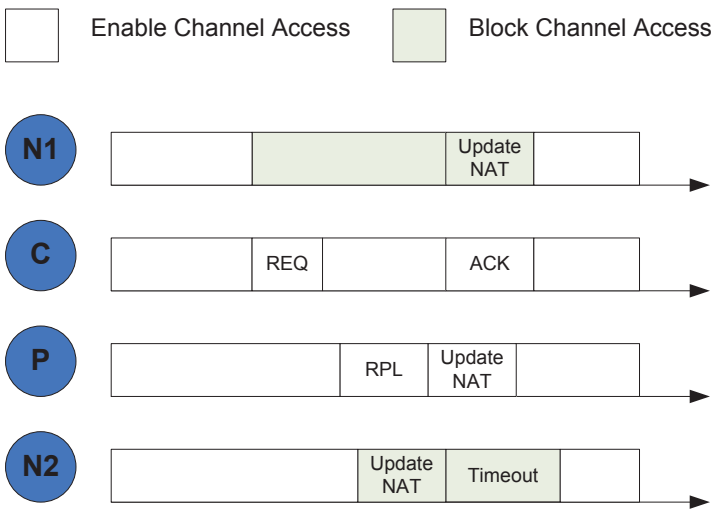

Fig. 3.4. Slot reservation process.

when a node negotiates slots with its parent, the slots occupied by their neighbors will not be reused.

Fig. 3.4 shows the aforementioned process where a child (C) tries to reserve a slot from the parent (P). The child's neighbor, N1, will refrain from channel access after eavesdropping of the REQ from C. N1 sets up a timer with the length equal to the time required for a complete REQ-RPL-ACK message exchange. N1 regains channel access when it receives ACK from C and updates NAT, or the timer expires. The P's neighbor, N2, will also be blocked from transmission when it receives RPL from P. N2 then updates the occupation of the newly assigned slot with default TTL in the NAT. It will also set a timeout which equals to the amount of time for RPL-ACK message exchange. It is then allowed to access the channel after the timeout. The block of channel access for N1 and N2 is to prevent concurrent slot reservation and assure that the process of reservation is conducted in sequence. Note that all the neighbors must cancel any pending slot request and regenerate new request since the NAT has been changed after resuming from channel access.

However, during the process if the parent finds the requested slot has been allocated to other node, it will refuse the slot assignation and respond with rejection (REJ), as shown in Fig. 3.5. Since no ACK message will be sent by C, N1 is blocked for channel access until

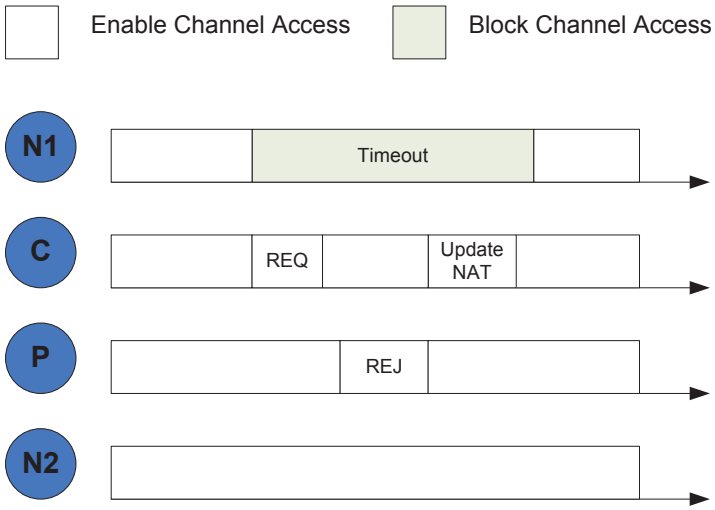

Fig. 3.5. Rejected reservation.

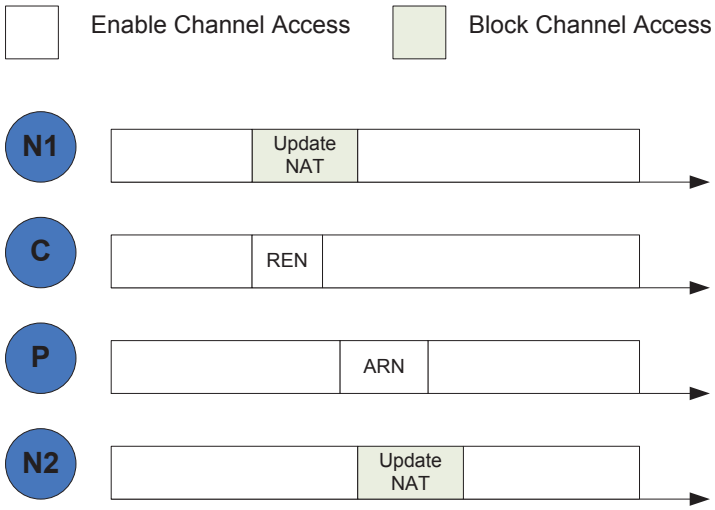

Fig. 3.6. Renew reservation.

timeout. N2 will not be affected and free to access the channel. The child will update the NAT according to the REJ message and start a new request process later.

Each node also needs to record its own TTL. When the TTL approaches zero, a node has to renew the reservation of time slots by sending a renew request (REN) to its parent. The parent nodes will then acknowledge the renewal (ARN) and update their NAT. Neighboring nodes will also update their NATs and extend the allocation duration. The renewing process is depicted in Fig. 3.6.

The aforementioned protocol provides nodes with the latest slot allocations of neighbors and their children and seamlessly embeds the information in the parent selection process. With the help of such a distributed slot scheduling protocol, a collision-free channel is assigned to each node and at the same time an upstream route is established for forwarding data packets from sensor nodes to the sink node.

3.3.4. Optimization of Schedules

Based on the design requirement, schedules assignation can be optimized for maximal concurrency, or minimum delay. A maximal concurrency schedule maximizes the set of concurrent transmitters and minimizes total number of unique time slots in the network so as to achieve higher throughput. It is similar to graph coloring problem has been proved to be an NP-Complete problem [42]. However, in a low rate monitoring network, throughput is not a primary concern and the total number of time slots is sufficiently large than the number of nodes. Therefore maximal concurrency scheduling is not suitable for this application.

On the other hand, a minimal delay schedule minimizes the multihop end-to-end delay. In [43], the authors formulate the wakeup scheduling as a graph-theoretical problem and show that minimizing the end-to-end communication delay in networks with arbitrary communication flows is in general NP-hard. However, if protocols only target at minimizing the upstream unicast delay, the complexity can be greatly reduced. [35] introduces a four-step algorithm to establish a minimal delay schedule. The centrally controlled algorithm requires gathering network-wise information and thus introduces large amount of overhead. In addition, topology of a wireless sensor network is usually dynamic due to the loss of nodes, channel interference or changing physical environment. Frequent rescheduling may be required to maintain an optimal schedule and result in considerable overhead. Therefore, current version of the slot scheduling protocol assigns TDMA slots randomly rather than optimize for minimum delay.

3.4. Time Synchronization Protocol

Time synchronization is an essential building block of sensor network systems [62, 67]. As discussed in Chapter 1,the applications envisioned for sensor networks vary from monitoring remote habitats and disaster areas to managing asset in warehouse and detecting and controlling machinery operation. Regardless of what they sense, one essential requirement is to know the timing of the observations. Therefore, sensor nodes usually maintain a clock in order to provide the timing information, necessitating global time synchronization in the network to ensure the clock consistency among distributed sensors [70, 45, 46, 44, 71].

In this research, the time synchronization module not only supports for timing measurement data, but also assists for the coordination of duty-cycle scheduling, as indicated in Fig. 3.1. In fact, the performance of fully synchronized MAC protocol heavily depends on the accuracy of time synchronization. Misalignment of slots will lead to large quantity of retransmissions, or even worse, no successful communication at all. The authors in [35] utilize a dedicated out-of-band synchronization hardware to achieve global synchronization. While very high accuracy can be obtained, such a method increases the complexity of the overall system design and requires a custom-made sensor platform. In this research, I found that in-band software-based synchronization schemes are completely practical and desirable accuracy for duty cycle scheduling can be obtained.

Design requirements of time synchronization methods are largely application-specific. Microsecond accuracy must be achieved for localization and tracking purposes [54], while synchronization error within the millisecond range is tolerable for coordinating slot scheduling. In particular, the design requirement of time synchronization protocol in this research is to minimize power consumption and communication overhead while maintaining millisecond accuracy.

3.4.1. Clock Architecture of Wireless Sensors

It is usually impractical (or too expensive) to equip every sensor node with a real-time clock. Instead, each sensor node maintains a local clock that is essentially an integer

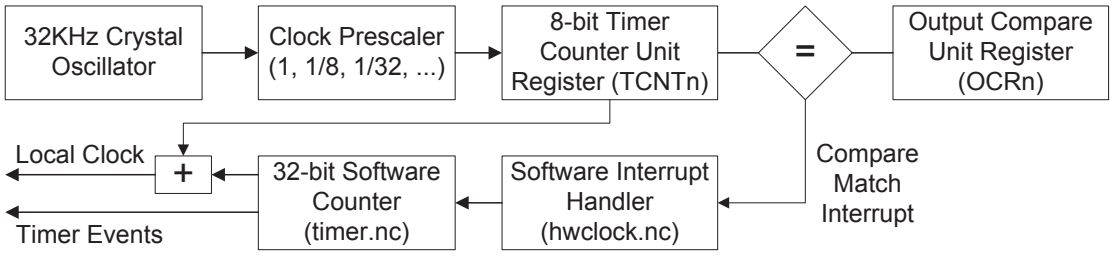

Fig. 3.7. Typical clock counter architecture of IRIS with ATmega1281 processor.

counter triggered by an external crystal oscillator. For example, Fig. 3.7 shows a typical configuration of the clock counter of the IRIS mote [59] with ATmega1281 micro controller unit (MCU) [74], a widely used low-cost sensor network system [67, 68]. The 32 kHz oscillator signal is first prescaled by dividing the frequency by a factor of 1, 8, 32, 64, 128, 256, or 1024 to configure hardware timer counter (HTC) resolution. The counter value in HTC will be incremented by 1 each time it receives a signal pulse. When HTC reaches a predefined value stored in the output compare unit (OCU), an interrupt will be generated and the software handler will increase the value of a 32-bit software timer counter (STC) by same amount as configured in OCU, instead of by 1, to preserve the time resolution of the HTC. HTC will be automatically cleared to 0 as it raises the interrupt. Thus, the clock time t of a sensor node is defined as

$$t = t_{\mathrm{HTC}} + t_{\mathrm{STC}}, \tag{1}$$

where t_{HTC} and t_{STC} are the current readings of the HTC and STC, respectively. The clock time t can be converted to a standard time unit based on the HTC's time resolution.

The timer interrupt handler and software counter are defined in hwclock.nc and timer.nc, the timer drivers in TinyOS. The drivers also create software timer events for many operations, such as waking up the MCU, scheduling a task to run and setting up CSMA backoff, etc.

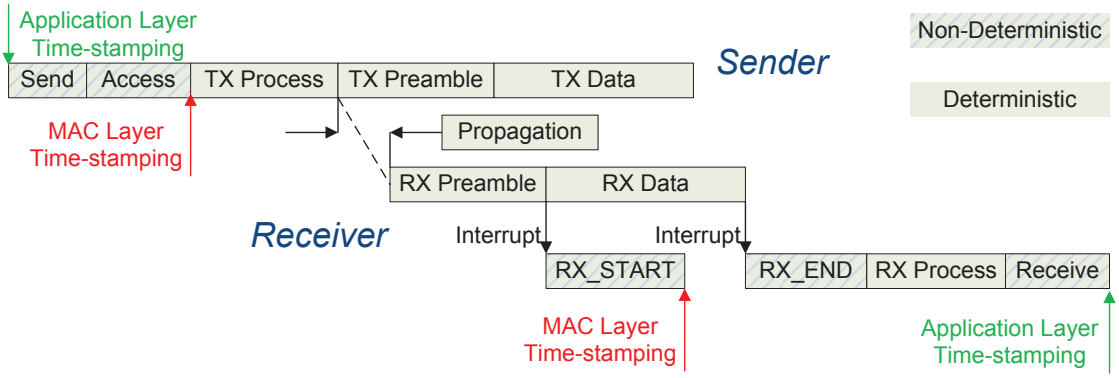

Fig. 3.8. Decomposition of message delay over a wireless link.

3.4.2. Delay Analysis in Radio Message Delivery

Synchronizing two sensor nodes requires the pair to exchange their current clock readings. The sender reads its current time and embedded the time-stamp in the synchronization message. The receiver also need to timestamp the message upon reception and record the timing information in the message as well before processing it. The message thus contains a pair of timestamps through which the clock discrepancy between the two nodes can be analyzed. However, the delivery of synchronization message inevitably experiences variable delays at both transmitter and receiver due to uncertain send, access, transmission, propagation and reception times [44]. These delays can be magnitudes larger than the required precision of time synchronization and consequently the pair of timestamps cannot be considered to be acquired at the same time. Instead, delays need to be carefully analyzed and compensated for. The decomposition and analysis of the sources of the message delivery delays were introduced in [44, 45, 46]. Fig. 3.8 shows the decomposition of packet delay when it traverses over a wireless link between two sensor nodes. Although the decomposition has been well presented in [44], the discussion was based upon former generation of Mica2 motes [59]. As the delays vary across platforms, I reevaluated the decomposition analysis and modified it according to the latest RF230 [73] radio transceiver.

Delays involved in the process can be classified as deterministic or non-deterministic. Deterministic delays are only parameterized by a fixed set of predictable factors such as

packet length and data rate and thus can be accurately estimated by experiments or simple calculations. Non-deterministic delays, on the other hand, are subject to change caused by random events and conditions. These delays are difficult to compensate and will contribute directly to synchronization error.

Send time: time used by a sender to construct a packet and pass the packet to MAC layer for processing. The send time is nondeterministic and is depending on the operating system (OS) system call overhead, the current load of the task scheduler and the processor.

Access time: delay incurred when waiting for channel access. This is specific to the design and implementation of MAC protocol in use. In this work, the hybrid MAC protocol will require the sender first to wait for its slot. And then during its transmitting slot, the sender will also need to perform random back off until the channel is clear. If a packet buffer or queue is employed, it may also include the waiting time in the queue, which can be as high as several minutes, depending on the actual throughput of the protocol.

Transmitter processing time: When the channel is clear, the MAC layer will pass the packet to the transceiver for further processing, such as adding physical layer headers and encoding the packet. The time used for sending packet from MCU to the radio chip usually takes several hundred microseconds, which can be calculated based on the interface data rate and packet length. The processing inside the transceiver typically requires 16μs [73].

Transmitting preamble time: Before transmitting the actual packet, a preamble consisting of four octets must be sent so that the receiver can detect a valid frame. The transmission of preamble takes 128μs [73].

Transmitting data time: The time for transmitting the actual packet over the wireless link. This delay is mainly deterministic in nature and can be estimated using the packet size and the radio speed. In the current platform, this time is typically in the order of a few milliseconds.

Propagation time: This is the actual time taken by the packet to traverse in the air from the sender to the receiver. It depends only on the distance between the two nodes and the absolute value of this delay is negligible as compared to other sources of packet latency.

Receiving preamble time: The receiver will spend the same amount time on receiving the preamble as the preamble transmission. It will take extra 8μs to process and detect the preamble.

RX start interrupt handling time: Upon successful detection of a preamble, the radio chip at the receiver will raise an interrupt to signal the start of receiving. It will take less than a few microseconds for the MCU to finish currently executed instruction and respond to the interrupt. Then the interrupt handler will take over the control of MCU and execute a few lines of code.

Receiving data time: While the MCU is responding to the receive start interrupt, the radio chip is busy receiving the data packet over the air. This time also matches with the transmitting data time, plus 16μs of processing overhead.

RX end interrupt handling time: After receiving the entire packet, the transceiver will generate an interrupt to notify the MCU for picking up. As with other interrupt handling time, this latency is non-deterministic but within a few microseconds.

Receiver processing time: It's the time for the MAC layer to read the received packet from the radio module. Similar to transmitter processing time, it usually requires several hundred microseconds.

Receive time: time to process the incoming message and to notify the receiver application. Its characteristics are similar to that of send time.

As discussed above, most of the delay uncertainties are introduced in the send time, access time and receive time. In fact, it is expected that the access time would completely overshadow other delays in practice. If the time synchronization messages are treated as normal messages, precise delay estimate is hard to obtain, if ever possible.

To alleviate the delay uncertainty, MAC layer time stamping was proposed, firstly in [46] and then improved in [44]. As depicted in Fig. 3.8, MAC layer time stamping enables the time synchronization protocol to modify its messages after passing it to the MAC layer. Note that the implementation the MAC layer time stamping is completely based on operation procedure of RF230 radio chip. At the sender side, time synchronization module can inject the latest time stamp into the message right before it will be delivered to the physical layer, i.e. radio chip. At the receiver side, the module can record time stamps as early as in the RX start interrupt handler. The MAC layer time-stamping effectively eliminate the majority of jitters during both transmission and reception. The remaining delays are mostly deterministic, and within millisecond range.

3.4.3. Related Works

Recently, a number of efficient synchronization protocols have been proposed including RBS [45], TPSN [46], and FTSP [44]. With the RBS approach, a beacon message is broadcasted by a beacon node and two sensor nodes synchronize between themselves by exchanging their local receiving times of the beacon. Thus, RBS eliminates transmitter-side uncertainty, although time-stamping at the lower layers of networking protocol stack may achieve the same effect. Such a method incurs a large communication overhead in large-scale networks due to pair-wise message exchange. The TPSN approach removes delay uncertainties at both sender and receiver through MAC layer time-stamping and it gains additional accuracy over RBS by averaging time-stamps of two messages. However, the two-way communication required in this method also results in a high communication overhead. The FTSP approach performs MAC layer time-stamping and it is able to synchronize multiple receivers with a single broadcast message. Such a flooding-based method is also insensitive to topological changes. Hence, a modified version of FTSP is adopted in the design due to its low overhead characteristic.

3.4.4. Modified FTSP

In the new design, similar to FTSP, each node maintains a buffer containing the latest time stamps for estimating clock skew and offset. The buffer window is also used for outlier detection to filter out corrupted time measurements. From several experimental studies, it is observed that FTSP can achieve less than 1ms timing errors in a three-hop network when the power management functionality is turned off. However, in low-power modes, FTSP results in errors of several hundred milliseconds. Through carefully examination of the timer driver shipped with FTSP, it is found that the driver is not able to return consistent time stamps in low power modes. In TinyOS, timer driver is required to implement two types of timer interfaces: one-shot timer and repeat timer. A one-shot timer fires only once whereas a repeat timer fires periodically until being called off. All existing timer drivers rely on a single hardware clock to handle both types of timers. During the sleep periods, clock is the only active module thus it dominates the power consumption in sleep mode. In a duty cycling system, the sleep mode power consumption decides the lower-bound of the average power consumption. Hence to save energy in low-power modes, existing timer drivers pull down clock frequency at which repeat timers request when there are no active one-shot timers. However, to support the one-shot timers, which usually fire in a few milliseconds, the clock has to run at high frequency, which leads to high central processing unit (CPU) usage and high power consumption. Unfortunately, switching between high and low frequencies results in inconsistent time stamps. Therefore, I developed a new two-layer timer driver to replace the original driver, which employs two individual hardware clocks to tackle the two types of timer separately. A high speed clock is used to drive one-shot timers, which remains active for a short period of time in normal mode. On the contrary, a low speed and thus low power clock runs continuously to support the repeat timers. Two clocks are synchronized from time to time to ensure consistency in time stamps.

3.5. Multihop Routing Protocol

A common characteristic of monitoring systems is, at most of the time, the data are flowing into a sink, the base station (BS) node. The BS node is usually equipped with another type of network interface, such as Wi-Fi, Ethernet adapters, or cellular modem. It functions as a gateway between the wireless sensor network and the Internet. Consequently, the network architecture could be described as a spanning tree structure rooted at the BS node. Any node that is one hop further away from the BS will need to pass data off to its parent node in the tree to report its readings. The multihop routing protocol is responsible for discovering neighbors, selecting parent nodes for routing and constructing and maintaining network topology.

3.5.1. Related Works

According to the operating modes, routing protocols can be classified as proactive or reactive [24]. Proactive protocols periodically monitor peer connectivity to ensure the ready availability of routing path for all nodes. Sensor nodes proactively announce their routing states to their neighbors, update routing paths to reflect topology changes, and transmit data according to a routing table. On the other hand, reactive protocols establish paths only upon request, e.g. in response to a query, or an event. No routing tables will be maintained and sensors remain idle in terms of routing behavior if no transmission of data. To query for some data, sensors forward each routing request to peers until it arrives at the destination. The requestee will respond over the reverse communication path.

Ad hoc on-demand distance vector (AODV) routing [48] is the default routing protocol for Zigbee networks [51], a popular wireless sensor network standard. As a reactive protocol, AODV protocol originates path requests on demand. Route requests are broadcasted to the entire network through multihop flooding. All intermediate relays compete with each other to become part of the best path. They increment the hop count of the packets and rebroadcast the request; meanwhile they set the sender as the next hop to the originator. Upon receipt of a request, the destined node establishes a path in reverse: they propagate

their reply using the previous sensor as the next relay to the originator. More than one request per source may be picked up by the requestee. Usually, the first request traverses the fastest path. The destined node can reinforce multiple paths to an originator. The needy node then begins using the route that has the least number of hops through other nodes. Unused entries in the routing tables are recycled after a time. When a link fails, a routing error is passed back to a transmitting node, and the process repeats. The established path is preserved until it expires implicitly or errors occur.

Reactive routing protocols have been the protocols of choice in mobile ad hoc networks, as they respond quickly to topology change, which may be very frequent due to node mobility. Due to their simplicity, and inherent support for data on demand, they have been the predominant design choice in mobile sensor networks. However, for environmental monitoring networks involving only stationary sensors to collect readings over time, proactive protocols are more efficient as the routing maintenance overhead could be distributed among a large number of data reporting through the same routes. Clustering protocols are examples of proactive routing. In a cluster-based architecture, nodes are organized into clusters. Each cluster elects a node as the cluster head for inter-cluster communication while other nodes mainly perform sensing task and can only communicate with the cluster head. Hierarchical routing is an efficient way to lower energy consumption within a cluster. By performing data aggregation and fusion, the number of transmitted packets to the BS is greatly decreased.

Low energy adaptive clustering hierarchy (LEACH) is a typical cluster-based protocol, which includes distributed cluster formation. LEACH randomly selects a few sensor nodes as cluster heads (CHs) and rotates this role to evenly distribute the energy load among the sensors in the network. The operation of LEACH is separated into two phases, the setup phase and the steady state phase. In the setup phase, the clusters are organized and CHs are selected. During the steady state phase, the sensor nodes can begin sensing and transmitting data to the cluster-heads. The cluster-head node, after receiving all the data, aggregates it

before sending it to the base-station to increase energy efficiency. After a certain time, the network goes back into the setup phase again and enters another round of selecting new CH.

Although clustering protocols such as LEACH are able to increase the network lifetime, they suffer from a serious limitation. They generally assume that all nodes can have sufficient and adjustable radio frequency (RF) transmission power and thus literally, all nodes are all within one-hop range. Therefore, it is not applicable to networks deployed in large regions. Their potentials could be fully utilized in a densely deployed sensor field, where nodes are located close to each other and have highly-correlated data.

Another family of proactive routing protocols is tree-based routing. Tree-based routing protocols organize the network into a spanning tree structure, rooted at the BS node. They effectively keep routing information and provide routing path in a simple manner, well suited for applications that collect data from all sensor nodes. Also, it is applicable for data fusion as parent nodes can aggregate data from their children nodes. In this research, I develop a tree-based routing protocol for multihop data collection. Combined with the hybrid MAC protocol described in previous section, the protocol provides energy-efficient and reliable data delivery, which is the essential requirement of the WSN.

3.5.2. Neighborhood Management

For proper operation, nodes manage a neighborhood table in which they store the information about the nodes they can hear from (literally their neighbors). There are two methods for neighbor nodes discovery. An active method for nodes to acquire such knowledge is to let them regularly broadcast probe messages, containing their ID, hop count and any other relevant information; all receivers of such packets may then add the sender to their table. Another method is to let nodes discover their neighborhood by overhearing packets introduced by other protocols. While this passive method reduces some control overhead, it will make the network layer protocol dependent on the messages generated by other protocols. Neighboring node discovery may fail if other protocols are unaware of such properties. Moreover, the information embedded in those packets is limited and reduce the performance

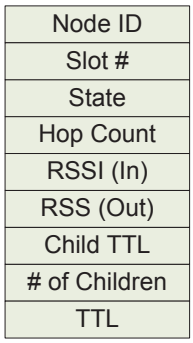

Fig. 3.9. Organization of a neighborhood table.

and functionality of the multihop protocol. Therefore the active probing is used in the protocol design. Periodically, each node will announce its state, hop count, slot number as well as received signal strength (RSS) from its neighbors. The neighborhood table maintains all the fields in the probe packets along with a TTL field. TTL will be updated upon receiving and successfully sending a packet, and decreased regularly or upon an unacknowledged transmission to the neighbor. Neighbors with low or zero TTL will be considered as dead and replaced by new nodes. The organization of a neighborhood table is portrayed in Fig. 3.9.

To determine the link quality of neighbors, each node is required to maintain estimates of both inbound (reception) RSS and outbound (transmission) RSS. Empirical studies [50] show that there are highly irregular links in real deployments of wireless sensor networks. Approximately 5% to 15% of all links are asymmetric links and asymmetric links vary significantly in different directions and distances. In addition, wireless sensor nodes may be programmed with different levels of transmission power or installed with antennas of unequal gains. All of the hardware, software and environmental factors may lead to asymmetric link and necessitate the link quality in both directions. In this design, bidirectional RSS estimates are kept in the neighborhood table and the less of them will be returned upon request for link quality.

3.5.3. Network Formation and Maintenance

A key attribute of a sensor network is its ability to self-organize. Every node has the intelligence to discover neighbors, measure radio signal strength and link quality with respect

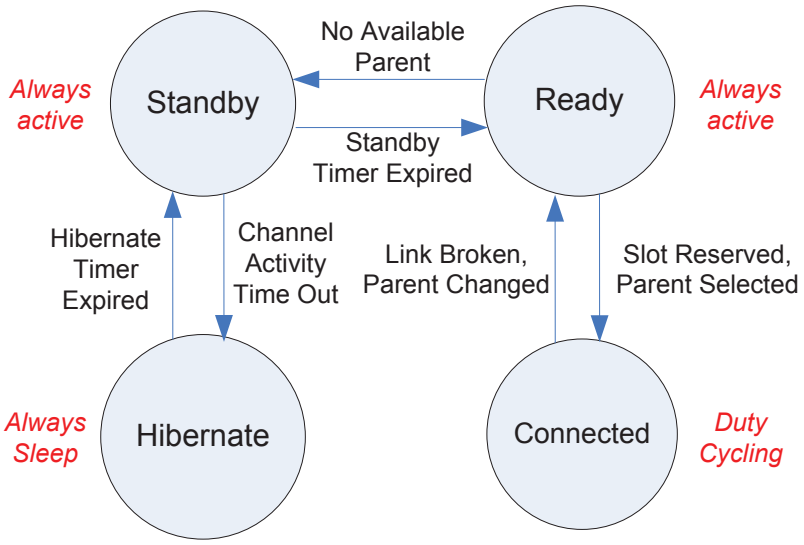

Fig. 3.10. The finite state machine used in a non-BS node.

to each neighbor, and establish a path to the BS node automatically in dynamic application environments. Thus, during the normal operation, the exact location of a sensor node in the spanning tree structure may change due to the varying radio propagation conditions.

Fig. 3.10 depicts the overall process how a new node joins an established network. An established network is defined as a subset of nodes, which have synchronized to the global clock and selected parents and slots for data reporting. Initially the subset contains only the BS Node. As mentioned in the previous section, nodes actively broadcast their current states at predefined interval in the contention slots. A node can easily identify the subset of nodes already in the network by searching its neighbors with READY state.

For a non-BS node, after power-up or reboot, it begins with the STANDBY state in which it is actively eavesdropping for channel activities to detect the existence of a network. The purpose of overhearing is to gather neighbor information such as slot occupation, neighbor link quality, and global clock stamp. This state lasts for a defined amount of time, generally longer than several super frames. To speed up the process, a node can also explicitly request for a certain type of information by broadcasting the requests. Then the node remains active and moves to READY state for the next step. However, if the node could not detect any channel activity during this period, the node will go into HIBERNATE state

and switch to sleep mode in order to conserve energy. The HIBERNATE state is very useful especially when the BS node fails. To join in the network, an offline node has to select one of its neighbors as the parent. The parent selection algorithm is described in later section. After reserving a TDMA slot successfully from the parent, the new node is CONNECTED to network and start duty cycle operation as described in previous sections.

The above process does not form a static network but a dynamic, self-healing network. A good link may become broken due to environmental conditions, new obstacles, unanticipated interferers and loss of individual nodes. When a node fails to report data to or receive any packet from its parent for several consecutive super frames, it will stop duty cycle and go back to READY state and try to reselect a parent.

3.5.4. Parent Selection

In order to join in the routing tree, a non-resident node is required to look for a parent as its next hop router for relaying data. The process begins with selecting a parent among neighboring nodes according to a parent selection algorithm detailed below. Once the parent is targeted, the joining node will reserve a TDMA slot from the parent based on the slot reservation protocol, as described in Section 3.3. In addition to updating its NAT table at the MAC layer, when accepting the slot reservation, the parent node will also set the ChildTTL field of the child node in the neighborhood table, as shown in Fig. 3.9. The nodes with ChildTTL greater than zero are recognized as children nodes. As with other TTL counter, ChildTTL will decrement across time. The parent will stop listening in the child's slot if the associated ChildTTL becomes zero. The ChildTTL will be also reset when a child renews its slot.

Many distance-vector based parent selection algorithms are available in the literature [47], which uses different cost metrics to guide routing. The cost of a node is an abstract measure of distance. In this protocol, the cost is parameterized by hop count, RSS, TTL, and duty cycle. When scheduled to run, the routing algorithm scans the neighbor list and find a potential parent with good link quality determined by the RSS and TTL, and

relatively low duty cycle. Among the parent candidates, the neighbor with lowest hop count is selected as the potential parent. If two or more candidates have the same hop count, the one with the lowest duty cycle will be selected so as to balance the routing load and maximize network lifetime.

CHAPTER 4

CLOCK ESTIMATION ALGORITHMS FOR WSN

Time synchronization in wireless sensor networks (WSN) has been studied extensively in recent years. However, the existing literatures are mostly focused on the time-stamp exchange protocol aspect of time synchronization whereas little emphasis has been put on the post-processing algorithmic aspect, especially from practical implementation perspectives. As introduced in Section 3.4, time synchronization in sensor networks generally begins with a message exchange protocol with which the nodes exchange their local time information. Then, the nodes calibrate their clocks to a common reference using some clock estimation algorithms. Reference broadcast synchronization (RBS) method [45], timing synchronization protocol for sensor networks (TPSN) [46], and flooding time synchronization protocol (FTSP) [44] are among the most widely cited time synchronization methods, or time synchronization protocols to be more accurate. In their original design, simple time synchronization algorithms are employed without in-depth analysis. In this chapter, I intend to fill the gap in the literature with a comprehensive study of clock estimation algorithms. More specifically, a unified formulation of least squares (LS) clock estimation algorithms to estimate clock offset, skew, and drift (see Section 4.1 for definition) using both batch and sequential estimators is presented. The simple algorithms used in RBS, TPSN, and FTSP are special cases of the ones presented in this research. For example, TPSN only accounts for clock offset and its algorithm can be viewed as a 0th-order batch estimator. Both RBS and FTSP utilize a 1st-order batch estimator to estimate clock offset and skew. However, it is important to note that, as discussed in the following sections, most of the algorithms presented in this chapter can be used together with the protocols of RBS, TPSN, and FTSP directly or with some minor adjustments.

Through extensive measurement and implementation experience, I have identified a number of key issues in the implementation of clock estimation methods, especially for continuous monitoring applications. In this chapter, I propose a suite of algorithms to address such issues, including a scaled signal model to achieve numerical stability in an ill-conditioned problem, sequential estimators for the scaled signal model to reduce computational complexity, a fast initialization scheme to improve energy efficiency, and outlier detection algorithms to improve robustness in long-term autonomous operations. The proposed algorithms are implemented in an actual WSN platform to demonstrate their practicality and to study their computational complexity empirically. Performance of the algorithms is studied through extensive measurement-based simulations with the measurement data collected in typical application scenarios.

4.1. Sources of Clock Synchronization Error in WSN

In sensor networks, each node has a clock counter that starts independently. The difference between the initial starting times is known as clock offset. Two oscillators may not run at an exactly same frequency, causing clock skew between two nodes. Oscillators suffer from aging effects and are affected by environmental variables, such as mechanical vibration, magnetic fields, and especially temperature [79]. Thus, the frequency of two oscillators may vary differently, causing clock drift between two nodes. If the clock offset θ_o, skew θ_s and drift θ_d coefficients between two nodes are known, time synchronization between a local time t and a reference time τ, which may or may not be the global or standard time, can be achieved through a 2nd-order polynomial time conversion formula,

$$t = \theta_o + \theta_s \tau + \theta_d \tau^2. \tag{2}$$

The choice of converting τ to t or t to τ is an application-specific design requirement; in some applications, both conversions may be required.

In continuous monitoring applications, duty cycling is often used to achieve energy-efficiency, which makes it desirable to reduce synchronization frequency. In order to maintain

synchronization over a long synchronization period, it may be necessary to estimate clock drift in addition to clock offset and skew, especially in harsh environmental conditions where oscillator frequency drifts significantly. In this chapter, various LS algorithms are derived to estimate clock offset, skew, and drift for the one-way broadcast-based synchronization model as employed in the RBS and FTSP approaches and study their performance based on measurement data.

4.2. Least Squares Estimation for Clock Estimation

This research adopts a simple broadcast-based model for time synchronization; that is, a reference node broadcasts a time-stamped synchronization message periodically and a client node time-stamps the received messages. The client node synchronizes its time t to the reference node's time τ based on a sequence of the time-stamp measurement data sets $\{\tau[i], t[i] : 1 \leq i \leq n\}$. In this section, several LS estimation algorithms are derived for such a synchronization model. The derivation is generalized for the p-th order polynomial estimator to suit both of the estimation of only clock offset and skew and the estimation of all three parameters.

4.2.1. Linear Least Squares Estimators

The signal model for the p-th order polynomial estimator at time n with w time-stamp measurement data sets, i.e., with a window size of w, can be defined as

$$\mathbf{x}[n] = \mathbf{H}[n]\theta + \mathbf{v}[n], \tag{3}$$

where the observation data $\mathbf{x}[n]$, the observation matrix $\mathbf{H}[n]$, and the observation noise $\mathbf{v}[n]$ at time n are defined as

$$\mathbf{x}[n] = [t[n-w+1] \ \cdots \ t[n]]^T,$$

$$\mathbf{H}[n] = [\mathbf{h}[n-w+1] \ \cdots \mathbf{h}[n]]^T$$

$$= \begin{bmatrix} 1 & \tau[n-w+1] & \cdots & \tau^p[n-w+1] \\ 1 & \tau[n-w+2] & \cdots & \tau^p[n-w+2] \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \tau[n] & \cdots & \tau^p[n] \end{bmatrix},$$

$$\mathbf{v}[n] = [v[n-w+1] \ \cdots \ v[n]]^T.$$

In practice, statistical characteristics of the observation noises are typically unknown a priori. Thus, the LS estimation method is desirable for such time synchronization problems, especially comparing to other model-based methods such as maximum likelihood (ML) and minimum mean squared error (MMSE) estimators. Derivation of LS estimators (LSE) does not rely on any probabilistic assumption of the observation noises and it is optimal in the sense that it is the best linear unbiased estimator (BLUE) for linear signal models where the observation noises are uncorrelated and have a zero mean and equal variances [76].

As discussed in Section 4.1, the time-stamp data $\tau[i]$ and $t[i]$ are 32-bit integer values. Thus, when $p > 1$, using the data directly in determining $\mathbf{H}[n]$ results in numerical overflow errors in computers. Converting the data to a standard time unit such as seconds cannot completely eliminate such overflow errors in practical implementation. Furthermore, given the form of $\mathbf{H}[n]$ in (3), computation of $(\mathbf{H}^T[n]\mathbf{H}[n])^{-1}$ directly or through Gaussian elimination to determine the LSE (similar to (8)) is an ill-conditioned problem, although well-posed. It is well known that if the elements of a matrix vary greatly in size, it is likely that large loss-of-significance errors will be introduced and the propagation of rounding errors will be worse [72]. To avoid such a problem, the matrix is generally scaled so that its elements vary less.

Following the discussion in [72], here I propose a scaling method to systematically address the numerical instability issue in clock estimation. Specifically, the data $\tau[i]$ and $t[i]$, $n-w+1 \leq i \leq n$, are scaled with $\tau[n]$ and $t[n]$, respectively, resulting in an equivalent

scaled signal model,

$$\tilde{\mathbf{x}}[n] = \tilde{\mathbf{H}}[n]\beta + \tilde{\mathbf{v}}[n], \tag{4}$$

where

$$\begin{aligned}
\tilde{\mathbf{x}}[n] &= \frac{1}{t[n]}\mathbf{x}[n] = [\frac{t[n-w+1]}{t[n]} \ \cdots \ 1]^T, \\
\tilde{\mathbf{H}}[n] &= [\tilde{\mathbf{h}}[n-w+1] \ \cdots \ \tilde{\mathbf{h}}[n]]^T = \mathbf{H}[n]\mathbf{S}[n], \\
\beta &= \frac{1}{t[n]}\mathbf{S}^{-1}[n]\theta, \\
\tilde{\mathbf{v}}[n] &= \frac{1}{t[n]}\mathbf{v}[n],
\end{aligned}$$

and the scaling matrix $\mathbf{S}[n]$ is a diagonal matrix and $\tilde{\mathbf{h}}[i]$, $n-w+1 \le i \le n$, are the vectors of $\mathbf{h}[i]$ scaled by $\mathbf{S}[n]$,

$$\mathbf{S}[n] = \text{diag}\{1 \ \frac{1}{\tau[n]} \ \cdots \ \frac{1}{\tau^p[n]}\}, \tag{5}$$

$$\tilde{\mathbf{h}}^T[i] = \mathbf{h}^T[i]\mathbf{S}[n] = [1 \ \frac{\tau[i]}{\tau[n]} \ \cdots \ (\frac{\tau[i]}{\tau[n]})^p]. \tag{6}$$

Then, by minimizing the LS error criterion

$$\tilde{J}[n] = (\tilde{\mathbf{x}}[n] - \tilde{\mathbf{H}}[n]\beta)^T(\tilde{\mathbf{x}}[n] - \tilde{\mathbf{H}}[n]\beta), \tag{7}$$

the standard linear LSE at time n can be derived for the scaled signal model as

$$\hat{\beta}[n] = (\tilde{\mathbf{H}}^T[n]\tilde{\mathbf{H}}[n])^{-1}\tilde{\mathbf{H}}^T[n]\tilde{\mathbf{x}}[n]. \tag{8}$$

With such an estimator, a set of w observation data samples $\{\tau[i], t[i] : n-w+1 \le i \le n\}$ are collected first and then processed all at once at time n. Thus, the estimator (8) is widely known as batch estimator, especially in contrast to the sequential estimator presented in the next section. With sequential estimators, at time n, only the new observation data sample $\{\tau[n], t[n]\}$ is processed to update the estimator derived at time $n-1$, without processing

any of the previous data $\{\tau[i], t[i] : i < n\}$; that is, the observation data sample is processed sequentially in time.

4.2.2. Sequential Least Squares Estimator

The sequential LSE for the unscaled signal model (3) can be found in [76]. Some simulation results of such an estimator can be found in [71], where the numerical stability issue in practical implementation is not considered. In this research the sequential LSE is derived for the scaled signal model (4), which can be directly employed in implementation of clock estimation algorithms. Benefits of the sequential estimator, as compared to the batch estimator (8), are widely known, including reduced computational complexity, lower memory requirement, and faster computation in real-time applications [76].

The sequential LSE for the scaled signal model (4) can be derived directly from the unscaled sequential LSE by incorporating scaling updates at every time steps. For the scaled signal model (4), the sequential estimator recursively determines the LSE of the unknown parameter $\hat{\beta}[n]$ and a covariance matrix (following the derivation in [76])

$$\tilde{\mathbf{\Sigma}}[n] = (\tilde{\mathbf{H}}^T[n]\mathbf{W}[n]\tilde{\mathbf{H}}[n])^{-1}, \tag{9}$$

where $\mathbf{W}[n]$ is a diagonal weighting matrix,

$$\mathbf{W}[n] = \text{diag}\{\frac{1}{\lambda}\ \frac{1}{\lambda^2}\ \cdots \frac{1}{\lambda^n}\}. \tag{10}$$

The parameter λ is a forgetting factor, $0 < \lambda < 1$, which is commonly used in sequential LS algorithms. By using the forgetting factor, previous data samples are exponentially down-weighted, effectively limiting the influence of the earlier data and allowing the estimator to react more quickly to model changes [76].

It is important to note that by using the sequential estimator, the estimation of $\hat{\beta}[n]$, $\tilde{\mathbf{\Sigma}}[n]$, and the minimum LS error $\tilde{J}_{\min}[n]$ is based on $\tilde{\mathbf{x}}[n]$ and $\tilde{\mathbf{H}}[n]$ with a window size $w = n$, i.e., the data $\tau[i]$ and $t[i]$, $1 \leq i \leq n$, scaled by $\tau[n]$ and $t[n]$ as in (4). To determine $\hat{\beta}[n]$, $\tilde{\mathbf{\Sigma}}[n]$, and $\tilde{J}_{\min}[n]$ recursively with the sequential estimator, we need to determine the estimators

of the unknown parameter $\hat{\beta}_n[n-1]$, the covariance $\tilde{\mathbf{\Sigma}}_n[n-1]$, and the minimum LS error $\tilde{J}_{\min,n}[n-1]$ at time $n-1$ that are derived based on the data $\tau[i]$ and $t[i]$, $1 \leq i \leq n-1$, but scaled by $\tau[n]$ and $t[n]$. It can be easily verified that

$$\hat{\beta}_n[n-1] = \frac{1}{t[n]}\mathbf{S}^{-1}[n]\hat{\theta}[n-1] \tag{11}$$
$$= \frac{t[n-1]}{t[n]}\mathbf{S}_n[n-1]\hat{\beta}[n-1],$$

$$\tilde{\mathbf{\Sigma}}_n[n-1] = \mathbf{S}^{-1}[n]\mathbf{\Sigma}[n-1]\mathbf{S}^{-1}[n] \tag{12}$$
$$= \mathbf{S}_n[n-1]\tilde{\mathbf{\Sigma}}[n-1]\mathbf{S}_n[n-1],$$
$$\tilde{J}_{\min,n}[n-1] = (\frac{t[n-1]}{t[n]})^2\tilde{J}_{\min}[n-1],$$

where

$$\mathbf{\Sigma}[n] = (\mathbf{H}^T[n]\mathbf{W}[n]\mathbf{H}[n])^{-1},$$
$$\mathbf{S}_n[n-1] = \mathbf{S}^{-1}[n]\mathbf{S}[n-1]$$
$$= \text{diag}\{1 \ \frac{\tau[n]}{\tau[n-1]} \ \cdots \ (\frac{\tau[n]}{\tau[n-1]})^p\}.$$

Thus, following the derivation for the unscaled signal model in [76], the sequential estimator for the scaled signal model (4) can be summarized as follows:

Scaling Update:

$$\hat{\beta}_n[n-1] = \frac{t[n-1]}{t[n]}\mathbf{S}_n[n-1]\hat{\beta}[n-1], \tag{13}$$
$$\tilde{\mathbf{\Sigma}}_n[n-1] = \mathbf{S}_n[n-1]\tilde{\mathbf{\Sigma}}[n-1]\mathbf{S}_n[n-1],$$
$$\tilde{J}_{\min,n}[n-1] = (\frac{t[n-1]}{t[n]})^2\tilde{J}_{\min}[n-1].$$

Estimator Update:

$$\hat{\beta}[n] = \hat{\beta}_n[n-1] + \tilde{\mathbf{K}}[n](1 - \tilde{\mathbf{h}}^T[n]\hat{\beta}_n[n-1]), \tag{14}$$

where

$$\tilde{\mathbf{K}}[n] = \frac{\tilde{\mathbf{\Sigma}}_n[n-1]\tilde{\mathbf{h}}[n]}{\lambda^n + \tilde{\mathbf{h}}^T[n]\tilde{\mathbf{\Sigma}}_n[n-1]\tilde{\mathbf{h}}[n]},$$
$$\tilde{\mathbf{h}}[n] = [1\ 1\ \cdots\ 1]^T.$$

Covariance Update:

$$\tilde{\mathbf{\Sigma}}[n] = (\mathbf{I} - \tilde{\mathbf{K}}[n]\tilde{\mathbf{h}}^T[n])\tilde{\mathbf{\Sigma}}_n[n-1]. \tag{15}$$

Minimum LS Error Update:

$$\tilde{J}_{\min}[n] = \tilde{J}_{\min,n}[n-1] + \frac{(1 - \tilde{\mathbf{h}}^T[n]\hat{\beta}_n[n-1])^2}{\lambda^n + \tilde{\mathbf{h}}^T[n]\tilde{\mathbf{\Sigma}}_n[n-1]\tilde{\mathbf{h}}[n]}. \tag{16}$$

It is important to note that no matrix inversions are required in the sequential algorithm, which significantly reduces computational complexity as compared to the batch estimator (8). To initialize the sequential algorithm, $\hat{\beta}[0]$ may be determined with a batch estimator using initial w data samples while $\tilde{J}_{\min}[0]$ and $\tilde{\mathbf{\Sigma}}[0]$ may be initialized with 0 and a large diagonal matrix, for example, $10^5\mathbf{I}$, respectively. We may also initialize with $\hat{\beta}[0] = \mathbf{0}$ [76]. After a burn-in period n_0, the estimator will converge with little biasing effect of the initial values. In long-term monitoring applications, the time index n increases indefinitely and thus λ^n in $\tilde{\mathbf{K}}[n]$ in (14) decreases indefinitely, causing numerical instability. Therefore, it is necessary to reinitialize the sequential algorithm periodically after n becomes too large, e.g., 10^3, by resetting n to 1, $\hat{\beta}[0]$ to the current estimate, $\tilde{J}_{\min}[0]$ to 0, and $\tilde{\mathbf{\Sigma}}[0]$ to $10^5\mathbf{I}$.

4.2.3. Energy-Efficient Fast Initialization Scheme

An initialization stage is required in both of the batch and sequential estimators; that is, with a window size of w, the batch estimator requires a collection of w data samples to start the estimation process while the sequential estimator requires an additional n_0 data samples for the burn-in period. In continuous monitoring applications, large data sampling

interval will make the initialization process very long. For example, assuming a duty-cycling period T of 15 minutes with 14 minutes of sleep period T_s and 1 minute of active period T_a, if the initialization stage is 10 data samples' long, sensor nodes need to stay awake for 150 minutes initially before starting duty-cycling with a synchronized time. Such a requirement puts a major burden on the energy resource of sensor nodes that are often powered by battery or solar cells in harsh environmental conditions. In this section, a novel initialization scheme is proposed to speed up the initialization process and significantly improve energy efficiency.

In designing the new scheme, it is important to note that data sampling for time synchronization does not need to be strictly periodic. The key idea of the new scheme is to use a small initial sampling period, then exponentially expand the sampling period until it reaches the regular duty-cycling period. Such a scheme makes it possible to start duty-cycling during the initialization stage to significantly reduce energy consumption. More specifically, a small initial sampling period T_0, e.g. 1 s, is employed at the beginning to collect $w + n_0$ data samples for initial estimation and burning-in. Then, the estimation algorithm continues with the sampling period increased by a times, e.g., 2 or 3 times. After the new sampling period is used to collect n_1 samples, it is increased again by a times to collect n_1 additional samples while the estimation algorithm continues. Such an exponential expansion of the sampling interval continues until the sampling period reaches the regular duty-cycling period and sensor node starts to follow the regular duty-cycling schedule. By gradually increasing the sampling period, sensor nodes are able to maintain synchronized time with adequate accuracy during the initialization stage. Thus, sensor nodes can start duty-cycling as soon as the current sampling period T_i is greater than the required active period T_a for data sampling; for example, stay active for T_a time units, then sleep for $T_i - T_a$ time units. The proposed method is referred to as the exponential expansion of sampling period (EESP) initialization scheme and the parameter a as the exponential expansion parameter.

Using the EESP initialization scheme, the time required to reach the regular duty-cycling schedule is

$$T_{\text{init}} = (w + n_0)T_0 + n_1 T_0 \frac{a^{m+1} - a}{a - 1}, \tag{17}$$

where

$$m = \text{round}(\log_a(T/a))$$

with the function round$(\cdot)$ denotes the rounding operation. If $T_a > T_0$, the total active time can be determined as

$$T_{\text{tat}} = (w + n_0)T_0 + n_1 T_0 \frac{a^{m_1+1} - a}{a - 1} + n_1(m - m_1)T_a, \tag{18}$$

where

$$m_1 = \lfloor \log_a(T_a/T_0) \rfloor$$

with the symbol $\lfloor \cdot \rfloor$ denotes the flooring operation; if $T_a \leq T_0$,

$$T_{\text{tat}} = (w + n_0)T_0 + n_1 T_a \frac{a^{m+1} - a}{a - 1}, \tag{19}$$

To better appreciate the benefits of the new initialization scheme, consider a simple example. Assume in a continuous monitoring application, $T = 15$ minutes, $T_a = 1$ minute, $(w + n_0) = 10$, $T_0 = 1\,\text{s}$, $a = 3$, $n_1 = 5$. Then, $T_{\text{init}} = 30.4$ minutes and $T_{\text{tat}} = 13.4$ minutes with the EESP scheme and 150 minutes without it. More comparison results are shown in Fig. 4.1. The advantage of the EESP scheme is more significant for the larger values of T and $(w + n_0)$. Therefore, the EESP scheme is strongly desirable for sequential estimators, especially when the burn-in period n_0 is large; with the EESP scheme, the extra cost of sequential estimators due to a long burn-in period, as compared to batch estimators, tend to be negligible in practice.

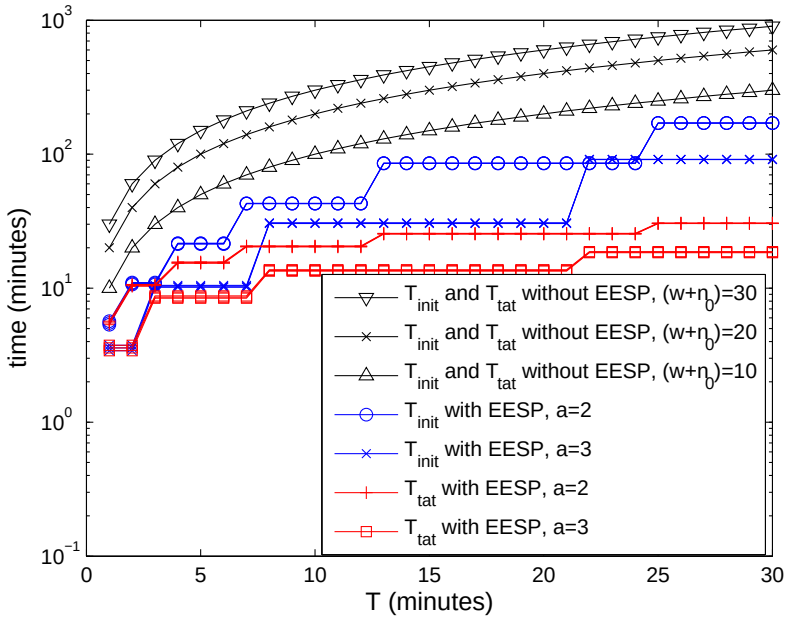

Fig. 4.1. Comparison of T_{init} and T_{tat} with and without EESP.

4.2.4. Outlier Detection

The synchronization message inevitably experiences variable delays at both transmitter and receiver due to uncertain send, access, transmission, propagation and reception times [44]. Due to the random nature of such delays, outliers may occur inevitably in time-stamp measurements. Outliers may also occur due to undetected hardware and software bugs that are impossible to avoid completely in implementation. Reading of the clock time needs to be an atomic operation, which must complete without anything else being able to change the value during the operation [78]. However, the clock time is a summation of two counters' readings as in (1), which is extremely difficult to implement as an atomic operation. Simply disabling interrupt does not solve such a problem; extra safeguard measures are needed to address all possible scenarios, which is an extremely difficult task in practice. In addition, reading hardware timer counter (HTC) shortly after wake-up may also give an incorrect result in some hardware platforms [74].

The outlier issue in time-stamp data must be effectively addressed, especially in duty cycling protocols. An incorrect time-stamp may result in misalignment of duty cycling

schedule, causing failure of entire network. Many outlier detection methods are available in the statistics literature, including methods based on global distribution, local distribution, distance, and deviation [75]. In this section a few simple distance-based algorithms that can be used together with the LSE in the preceding sections is presented.

An iterative minimum residual (IMR) algorithm is proposed in [77] to detect and reject outliers by iteratively searching for the minimum residual estimator among the LSE derived from different combinations of the data. With the IMR algorithm, outliers are eliminated one-by-one in an iterative way. For example, given a set of w data, first the batch LSE is derived based on all of the data. Second, w batch LSE is derived based on all possible combinations of the data, taking $w-1$ data at a time. Then, determine the best estimator in terms of minimum root-mean-square (RMS) LS residual error. The data not employed in the derivation of the best estimator is eliminated from the data set. This process continues iteratively with the reduced set of data until a certain criterion is met; for example, iterations have been conducted for a predefined number or the change in the minimum RMS residual error is less than a predefined tolerance δ, comparing to the previous iteration. Such an IMR algorithm can be used in the initialization stage of both batch and sequential estimators to reject outliers in the very first w data samples.

Starting from the $(w+1)$-th data, only the latest data may be examined sequentially for outlier detection as explained in the following. With the LSE at time n, $\hat{\beta}[n]$, an estimate of $t[n+1]$ may be determined for a given $\tau[n+1]$ from

$$\hat{t}[n+1] = t[n]\tilde{\mathbf{h}}_n^T[n+1]\hat{\beta}[n], \tag{20}$$

where $\tilde{\mathbf{h}}_n[n+1]$ is the vector $\mathbf{h}[n+1]$ scaled by $\mathbf{S}[n]$,

$$\begin{aligned}\tilde{\mathbf{h}}_n^T[n+1] &= \mathbf{h}^T[n+1]\mathbf{S}[n] \\ &= [1 \ \frac{\tau[n+1]}{\tau[n]} \ \cdots \ (\frac{\tau[n+1]}{\tau[n]})^p].\end{aligned}$$

The prediction error at time $n+1$ for the scaled signal model is then defined as

$$\epsilon[n+1] = t[n+1] - \hat{t}[n+1]. \tag{21}$$

The data $\{\tau[n+1],\, t[n+1]\}$ is considered an outlier if

$$|\epsilon[n+1]| \geq \min\{\epsilon_u,\, \max\{\epsilon_l,\, \epsilon_{\text{th}}[n]\}\}, \tag{22}$$

where the threshold $\epsilon_{\text{th}}[n]$ may be defined as three times or more of the RMS residual error at time n, $\epsilon_{\text{rms}}[n]$, while ϵ_l and ϵ_u are the empirical lower and upper bounds of the residual errors, respectively, employed to improve the robustness of the algorithm in practical implementation. When batch estimators are used, the RMS residual error can be determined at every time step as

$$\epsilon_{\text{rms}}[n] = t[n]\sqrt{\frac{1}{w}\tilde{J}_{\min}[n]}. \tag{23}$$

With the forgetting factor λ, the equivalent LS error criterion of sequential estimators is weighted by the diagonal weighting matrix (10) [76]. Thus, the RMS residual error of the sequential estimator can be determined as

$$\epsilon_{\text{rms}}[n] = t[n]\sqrt{\frac{\lambda^n(1-\lambda)}{1-\lambda^n}\tilde{J}_{\min}[n]}. \tag{24}$$

In summary, a pseudo code of the LS clock estimation algorithm is provided in Algorithm 1, which integrates the batch and sequential LSE, EESP, and outlier detection algorithms presented in this chapter.

4.3. Implementation and Computational Complexity

In order to study the practicality of the algorithms presented in this chapter, I implemented the LS clock estimation algorithms together with a modified version of FTSP [44] in the IRIS motes [59]. In particular, I employ the medium access control (MAC) layer time-stamping method and the timing message exchange protocol of FTSP, but replace the

Algorithm 1 The LS clock estimation algorithm with EESP and outlier detection at client node at time n

Sample the current time-stamp data $\{\tau[n], t[n]\}$.
if $n < w$ **then**
  Do nothing here.
end if
if $n = w$ **then**
  Use IMR algorithm to detect and eliminate outlier.
  Determine a batch estimator $\hat{\beta}[n]$.
end if
if $n > w$ **then**
  Use (22) to detect outlier sequentially.
  if $\{\tau[n], t[n]\}$ is outlier **then**
    Eliminate $\{\tau[n], t[n]\}$ from data sequence.
  else
    Determine a sequential estimator $\hat{\beta}[n]$.
  end if
end if
Use EESP to determine a new sampling interval.

native clock offset and skew estimation algorithm with the batch and sequential estimators described in Section 4.2. The experimental results presented in this section is obtained in a simple experimental setup consisting of a pair of motes, a parent node and a child node. The two nodes, placed 5 m apart in an outdoor experimental site, run the modified FTSP protocol to synchronize the child's time to the parent's. A third data collection node, deployed near the pair, sends a clock inquiry message every 30 seconds; both parent and child time-stamp the arrival time of the message and report the time-stamps to the data collection node. The differences between the time stamps of each query, namely the clock estimation errors between the parent and child nodes, are calculated and stored in the data collection node's flash memory for postprocessing.

The compiler (i.e., avr-gcc) for the ATmega1281 microcontroller (MCU) on IRIS only supports 32-bit single-precision floating-point computation. In the IEEE 754 standard, a single-precision binary floating-point number has 23-bit fraction component; however, the time stamps are 32-bit integers and the conversion from integer to single-precision floating-point number results in severe precision loss, especially when time stamps are large numbers.

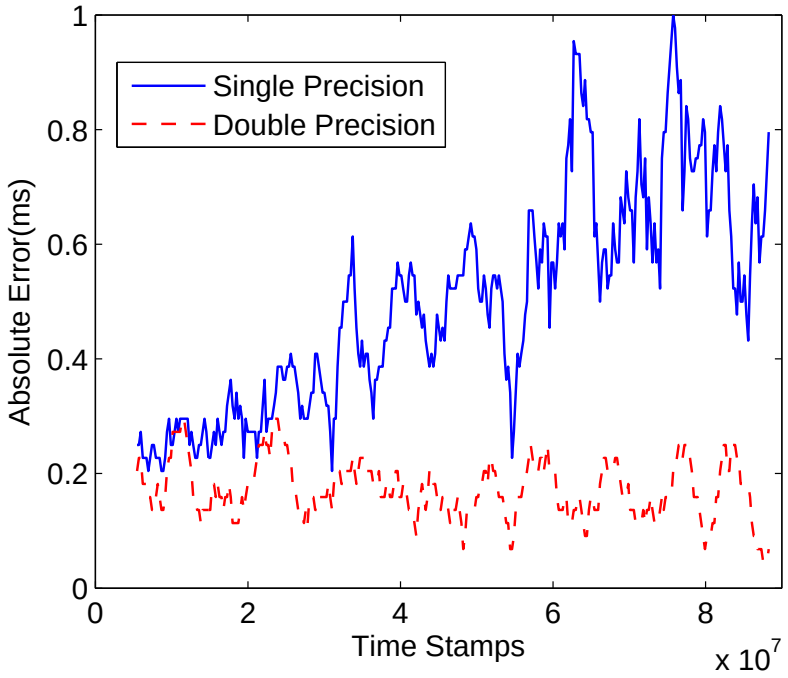

Fig. 4.2. Estimation errors of the 1st-order batch estimator, implemented in the IRIS mote.

To address such a problem, I implemented 64-bit double-precision floating-point operations in the IRIS platform. It should be noted that the more powerful (but less energy efficient) Imote2 mote [59] does not suffer from the precision loss problem as it has a built-in floating-point co-processor. Fig. 4.2 shows the estimation errors of the 1st-order batch estimator with both single- and double-precision floating-point computations, implemented in IRIS. With a synchronization sampling period of 1 min, the time stamps grow from 0 to 9×10^7 in 6 hours. It can be clearly observed that the increasing trend of estimation errors with the single-precision implementation. The 2nd-order estimators even fail to yield any valid result due to the divide-by-zero errors when implemented in single-precision floating-point.

To assess the computational complexity of the LS clock estimation algorithms, I have transplanted the code segments of the algorithms from TinyOS to C and evaluated the execution time of the C program in AVR studio, which is an integrated development environment (IDE) for writing and debugging programs for Atmel MCUs including ATmega1281. The C program is first compiled with the avr-gcc compiler and the machine code is then executed on

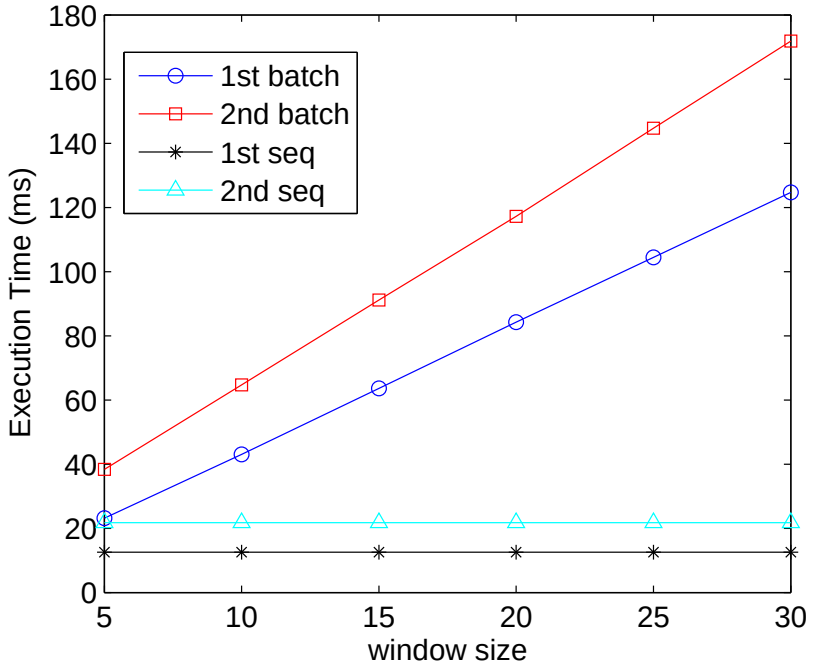

Fig. 4.3. Execution time of the LS clock estimation algorithms measured in AVR simulator.

the AVR simulator, which is a part of AVR studio. Unlike the high-level network simulators such as NS2 and TOSSIM that are insufficient to measure the execution time of a task or a code block, AVR simulator is able to simulate the program instruction-by-instruction and the clock cycle of each instruction can be precisely recorded. When simulating the program, I set up a watch timer at a breakpoint before entering the function of an algorithm. After the simulator exits the function block and stops at another breakpoint, the watch timer records the execution time between the breakpoints.

Fig. 4.3 compares the computation time of the four estimators. From the results, we can observe that the computation time of batch estimators is a linear function of window size whereas the execution time of sequential estimators is constant, independent of the forgetting factor (and window size), since only one time stamp is processed at a time with such estimators. On the other hand, the computation time of the 2nd-order estimators is only slightly higher than the 1st-order estimators. Computational complexity of an algorithm is directly related to its energy efficiency, which is one of the major design considerations of

resource-constrained WSN. The energy cost of each estimator can be obtained by multiplying the execution time by the average power consumption of a specific MCU, which is typically provided by manufacturers in terms of average current draw and operating voltage of the MCU.

In the next two sections, a comprehensive performance evaluation of the LS clock estimation algorithms using extensive measurement-based simulation results is present.

4.4. Measurement and Data Collection System

To study the behavior of the clock of typical sensor nodes, I have conducted extensive measurements using a simple experimental setup. The sensor network platforms used in this research are the IRIS (and MICAz) motes from Crossbow Technology, which are the most widely used low-cost sensor network platforms nowadays [26, 67, 68]. The measurement data are used in various simulations to evaluate the performance of clock estimation algorithms in realistic scenarios.

The measurement and data collection system consists of three sensor nodes, among which one node periodically broadcasts beacon messages while the other two nodes receive the messages and record their arrival time. In the implementation, the MAC-layer time-stamping technique is employed to eliminate the uncertainty in the reception time. All sensor nodes are stationary during measurement. Time-stamp data are saved in the local flash memory; data are retrieved through serial port after measurement. The time-stamp data are indexed by the sequence number of the beacon messages. In the post-processing stage, the time synchronization process between two sensor nodes is simulated using the time-stamp data collected using the measurement system; that is, the time-stamp data from one receiving node is used as the time τ of the reference node and the data from the other receiving node as the time t of the client node. Then, a clock estimation algorithm can be applied to the data to synchronize the time of the client node to the reference node's as presented in Section 4.2. Such a measurement-based simulation method can be used to conveniently evaluate the performance of clock estimation algorithms employed in a wide range of broadcast-based

time synchronization methods such as RBS and FTSP. Furthermore, such a measurement-based simulation method is especially useful when comparing the performance of different algorithms since the same set of measurement data can be used repeatedly for all algorithms, which ensures the fairness of the comparison.

As discussed in Section 4.1, the highest HTC time resolution may be achieved in the IRIS mote by setting the prescaling factor to 1, with which a time synchronization accuracy in the order of a few microseconds can be achieved. However, energy consumption of sensor nodes in sleep mode is directly related to the HTC time resolution since the 8-bit HTC interrupts faster with a higher time resolution. Thus, there is a trade-off between time synchronization accuracy and energy-efficiency. Targeting at low-power continuous monitoring applications such as environmental monitoring, here a prescaling factor of 8 that results in a HTC time resolution of 0.25 ms is employed. In the measurement system, the broadcasting period of the beacon messages, i.e., the sampling period of the measurement system, is set to 4 s. Thus, with a 128 kB flash memory on each node, I am able to collect about 10^5 data samples continuously for up to 11 hours. With such a sequence of measurement data, a new data sequence of lower sampling rate can be constructed in the post-processing stage through proper decimation.

Frequency drifts and aging effects of oscillators are affected by environmental conditions [79]. Thus, here I consider three typical scenarios, including indoor same condition (ISC), outdoor same condition (OSC), and outdoor different condition (ODC). In the ISC, all sensor nodes are placed inside an air-conditioned building. In the OSC, all sensor nodes are placed directly under the sun without any shade. In the ODC, sensor nodes are placed in an environment where surrounding trees form different dynamic shading patterns for the nodes with the sun movement. Outdoor measurements are started in the afternoon to capture data during both daytime and nighttime. To ensure the quality of the measurement data, all experiments are repeated several times with different combinations of motes, selecting 3 motes from a pool of about 20, and the data from the repeated measurements are cross-checked

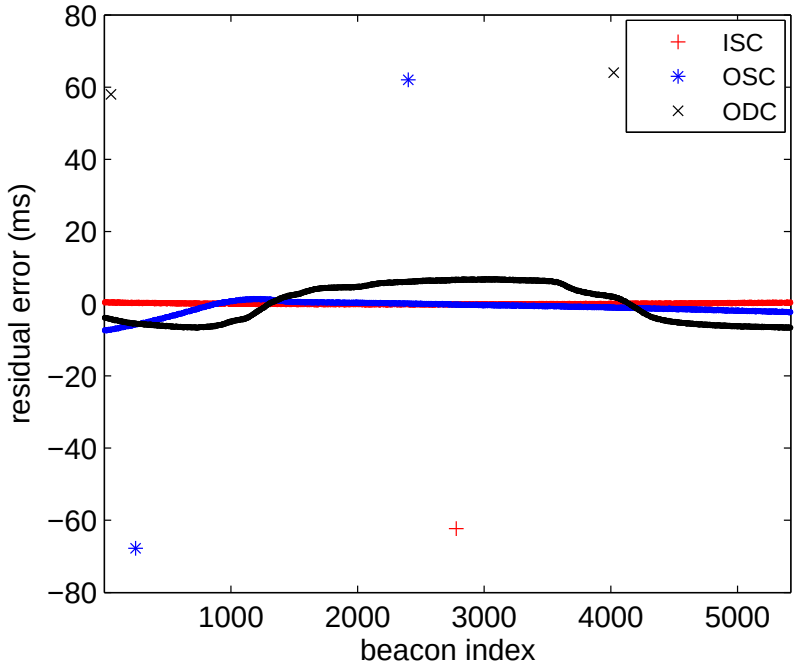

Fig. 4.4. Residual errors of three measurement data sequences after linear regression with a straight line.

carefully for consistency. Through such a process, I indeed found and eliminated some abnormal data sets caused by hardware anomalies in some of the motes. Sample measurement data are shown in Fig. 4.4. To better present the nonlinear clock drifting effect, the figure only plots the residual errors of each data sequence after linear regression with a straight line. The ODC scenario shows highest drifting effect as expected. Also, outliers are evident in the data, which is quite common in motes as discussed in Section 4.2.4.

4.5. Measurement-based Simulation Results

In this section, I study the performance of the proposed clock estimation algorithms through a number of measurement-based simulation results. The parameters of EESP and outlier detection algorithms are tuned through extensive simulations; for example, burn-in period $n_0 = 30$, exponential expansion parameter $a = 3$, the number of samples collected with each intermediate sampling period $n_1 = 5$, initial sampling period $T_0 = 4\,\mathrm{s}$ (which is determined by the measurement system configuration), outlier detection lower bound

$\epsilon_l = 8\,\text{ms}$ and upper bound $\epsilon_u = 48\,\text{ms}$. In addition, in the simulations, I use window size $w = 10$, forgetting factor $\lambda = 0.8$, and regular sampling period $T = 5$ minutes, if not otherwise specified. Various clock estimation algorithms are compared in terms of their root-mean-squared prediction error (RMSE) performance, which is determined as the root-mean-squared (RMS) value of the prediction errors, defined in (21), over the measurement data sequence.

4.5.1. Performance of EESP Algorithm

To study the effectiveness of the EESP algorithm, I conduct a series of simulations in all three measurement scenarios. The RMSE performances of the LS estimators in the initialization stage are presented in Fig. 4.5. From the results we can clearly observe that all estimators are quite insensitive to the parameter a, although a faster expansion tends to introduce larger estimation errors as expected intuitively. For example, even with 7 times expansion of the sampling period at each step, all estimators achieve less than 1 ms RMSE, which may well be an adequate performance to enable duty-cycling in many practical applications.

It is interesting to note that the 1st-order estimators have better performance than the 2nd-order ones. Such an observation is due to the fact that clock drifting effects are negligible within a short period of time; only over a long time period, clock drifting effects are significant. On average, sampling intervals are very small during the initialization stage. Thus, samples within a short window (e.g., $w = 10$ in the simulations) are more accurately modeled with the 1st-order polynomial model. In addition, the performances of batch and sequential estimators are comparable; the slight difference between them is because of the use of a fixed window size in batch estimators and a forgetting factor in sequential estimators to down-weight the previous data samples.

4.5.2. Effects of Sampling Period

Sampling period is an important system configuration parameter in practical implementations, especially with duty-cycling. In duty-cycling protocols, the sampling period

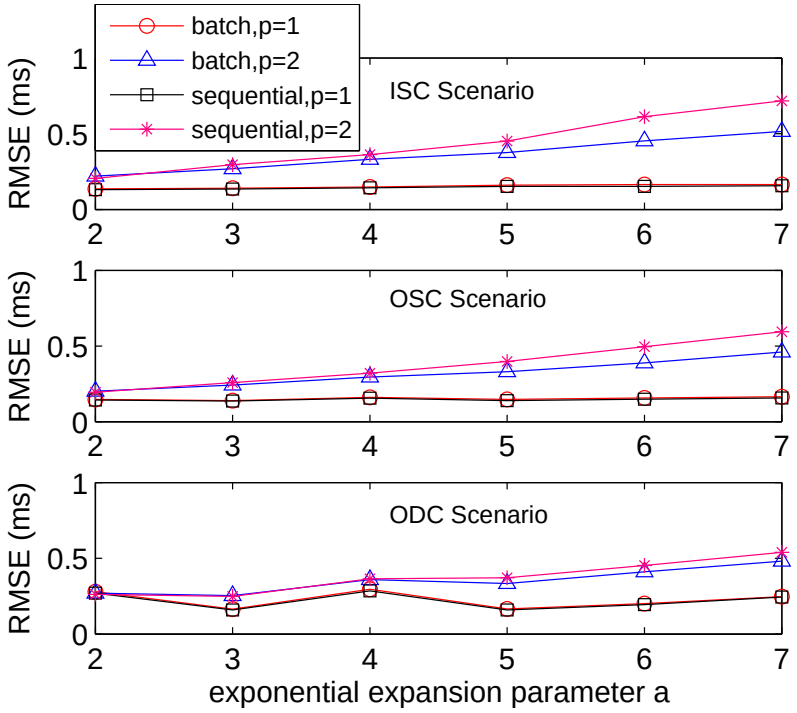

Fig. 4.5. Performance of the EESP algorithm versus parameter a.

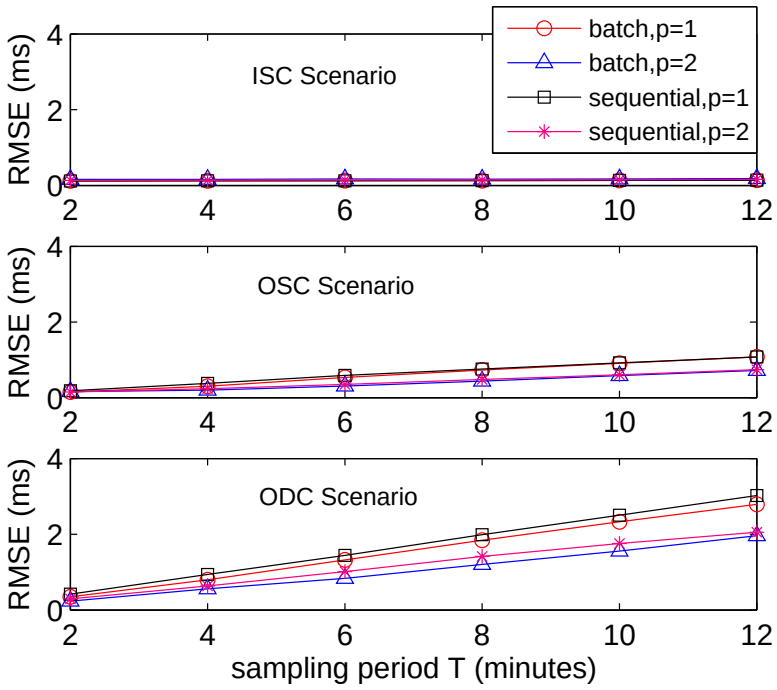

Fig. 4.6. Performance of the LSE versus sampling period.

determines how often a sensor node needs to wake up for synchronization purposes, which directly relates to the energy efficiency of the system. Performances of the estimators versus sampling period are shown in Fig. 4.6. In the ISC scenario, the performance of all four estimators are very close to each other and the sampling period does not have significant effect. However, in outdoor scenarios, especially in ODC, performances of all four estimators deteriorate steadily as sampling period increases. Such an observation is easily justified intuitively, considering that there are very minimum clock drifting effects in controlled indoor environments while in random outdoor deployments, clock frequency drifts much more significantly, especially over a long period of time. Thus, there is a trade-off between time synchronization accuracy and energy efficiency; in practice, the sampling period may be maximized subject to a design requirement of time synchronization accuracy.

From the results, we can also observe that the 2nd-order estimators have better performance than the 1st-order estimators when the sampling period is larger, especially in the ODC scenario, justifying the use of the higher order estimators.

4.5.3. Effects of Window Size and Forgetting Factor

To study the effects of window size and forgetting factor, I present the RMSE performances of the 2nd-order estimators in Fig. 4.7 and Fig. 4.8. Similar to the preceding discussion, we can observe that in the ISC scenario, the performance of the estimators tends to improve, although slightly, as window size or forgetting factor increases because of the inherent linearity of the clock behavior. However, a smaller window size or forgetting factor is preferred in outdoor scenarios.

By comparing Fig. 4.7 and Fig. 4.8, we can also conclude that when the sampling period is smaller, consecutive data samples are more closely correlated and thus a little larger window size is preferred; for example, the best overall performance is achieved with $w = 5$ and $\lambda = 0.7$ when $T = 5$ minutes, but with $w = 10$ and $\lambda = 0.8$ when $T = 1$ minute.

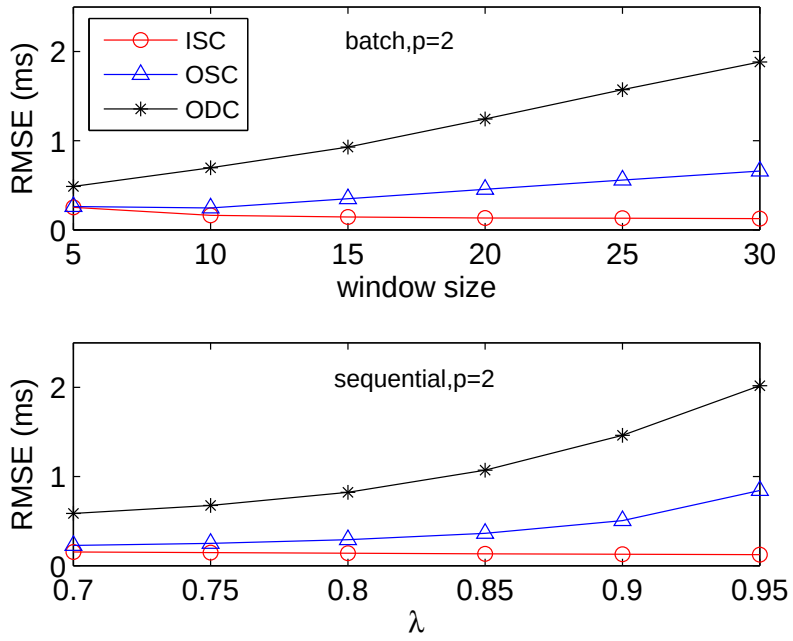

Fig. 4.7. Performance of the 2nd-order LSE with $T = 5$ minutes.

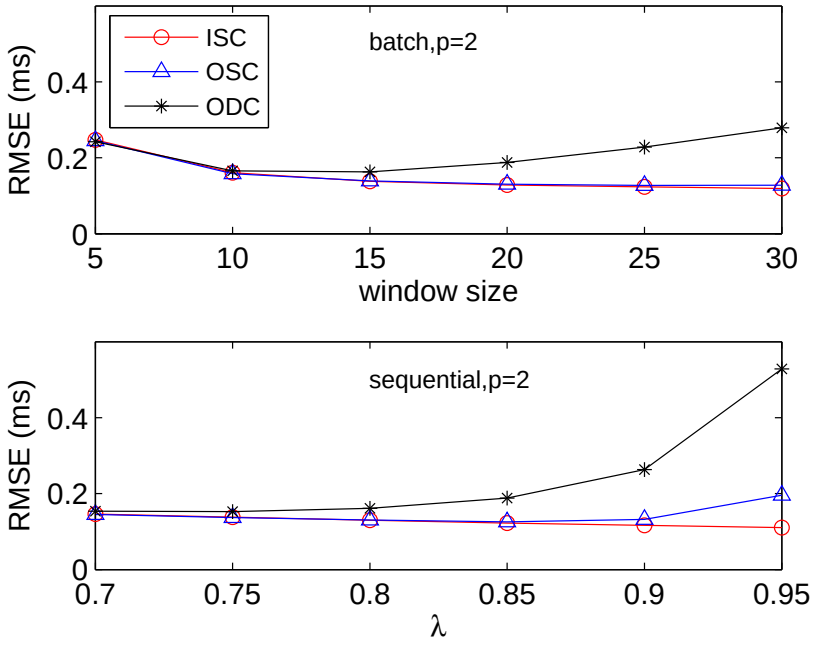

Fig. 4.8. Performance of the 2nd-order LSE with $T = 1$ minute.

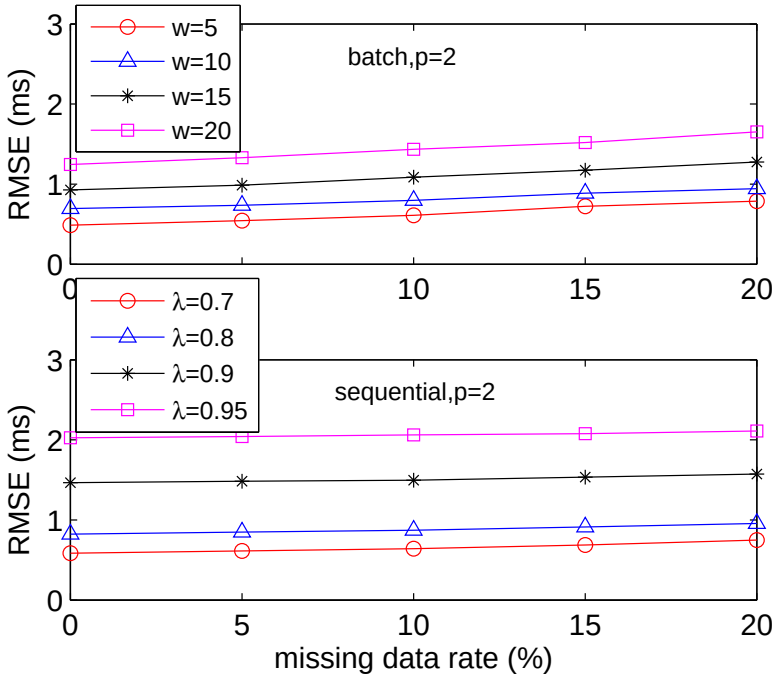

Fig. 4.9. Performance of the 2nd-order LSE with different missing data rate when $T = 5$ minutes.

4.5.4. Effects of Missing Data

To study the robustness of the proposed algorithms in realistic lossy communication conditions, I conduct a series of simulations of the 2nd-order estimators in the ODC scenario by varying the missing data rate. When the data is missing at time n, the latest estimator at time $n-1$ will be used to predict the time at the next time step $n+1$. From the results shown in Fig. 4.9 and Fig. 4.10, we can clearly observe that the estimators are remarkably robust in missing data conditions. For example, even with a missing data rate of 20%, the performance degradation is quite insignificant in all simulation configurations. In addition, even with missing data, a smaller window size or forgetting factor is preferred when the sampling period is relatively large, e.g., $T = 5$ minutes, as shown in Fig. 4.10. Better performance can be achieved with a larger window size or forgetting factor when the sampling period is smaller, e.g., $T = 1$ minute, as shown in Fig. 4.10.

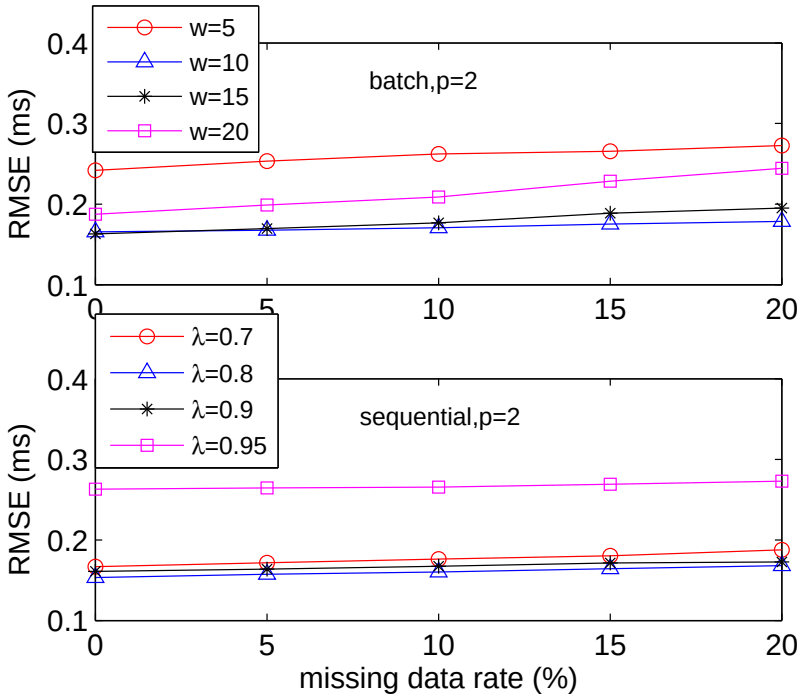

Fig. 4.10. Performance of the 2nd-order LSE with different missing data rate when $T = 1$ minutes.

4.6. Conclusions

In this chapter, a set of LSE and related algorithms are derived for clock estimation in continuous monitoring sensor network systems. The measurement-based simulation method employed in this research makes it convenient to study the performance of clock estimation algorithms in realistic scenarios. Simulation results have demonstrated the effectiveness of the proposed algorithms. In particular, I conclude that 1) the 2nd-order estimators are preferred in outdoor random deployment scenarios due to significant clock drifting effects that are inherently nonlinear over a long period of time, 2) the sequential estimators are able to achieve comparable performances to the batch estimators with reduced computational cost, and 3) small sampling period, and small window size and forgetting factor are preferred in outdoor conditions, especially when the sampling period is large.

CHAPTER 5

SYSTEM DEPLOYMENT AND EVALUATION

Unlike many existing works which evaluate the designs through simulations or short-term experiments, this research has implemented the proposed protocols and algorithms in realistic wireless sensor network (WSN) platform and incorporated the WSN system into a publicly available cyberinfrastructure, supporting long-term hydrologic monitoring and modeling research. This chapter details the implementation of the WSN-based soil moisture monitoring system, and shares the experience in the process of deployment and maintenance. Field results demonstrating the energy efficiency, reliability and autonomy are presented. The results are in fact part of the system status data, which are also collected together with the environmental sensor data, to facilitate remote infrastructure monitoring and maintenance.

5.1. WSN Implementation

I implemented the networking protocols and several sensing tasks on the IRIS mote from Crossbow Technology [59]. The IRIS mote provides a highly integrated, cost-effective hardware solution for low-power WSN applications. Based on the IRIS platform, Crossbow Technology also provides a series of interface boards and sensors boards to support difference functionalities. Three types of nodes are used in the system, namely base station node, sensor node and relay node. The base station (BS) node is an IRIS mote installed on an extension board MIB510 [59], which interfaces with the remote field gateway (RFG) server through the RS-232 serial port in the weather station. The data collected by other nodes are periodically transmitted to the BS node through multihop communications. Then, the BS node transmits aggregated data to the RFG server through serial port. Sensor nodes are made of an IRIS mote and an MDA300 data acquisition board. The MDA300 has 7 single-ended 12bit A/D

Fig. 5.1. Wireless sensor nodes in a soil sampling site.

channels for sampling external analog sensors, and 2.5V reference voltage for precise sensor excitation, which is crucial to ensure the fidelity of measurements. To support soil moisture monitoring, each MDA300 board is wired with one or several EC-5 probes [61], deployed at different depths at the designated soil sites. Since the soil sites are relatively far away from each other and from the weather station, I use bare IRIS motes as the relay nodes in order to increase the network coverage and connect all sampling sites.

To survive extreme weather conditions, the motes are installed in weatherproof boxes, and the boxes are installed 4 feet above the ground on top of metal poles to avoid flooding water. To improve wireless signal propagation, I have replaced the built-in antenna with high-gain (7dBi) external antenna. Fig. 5.1 illustrates three sensor nodes working in a soil sampling site. Each of the nodes encompasses an IRIS mote and an MDA300 board interfacing with three EC-5 probes, buried under the ground at different depths.

5.2. System Deployment

The Greenbelt Corridor (GBC) weather station in Denton, Texas, has been operational since 1999 with temperature, solar radiation, rain gauge, wind speed and direction, and soil moisture sensors, all of which are connected by wire to a datalogger and are deployed inside a small fence-enclosed area. However, the weather station can only provide a single observation point thus data collected by the station is not sufficient to characterize the physical and chemical conditions of the entire area of interests. To support long-term hydrologic monitoring and modeling in the floodplain area, the weather station system has been upgraded and integrated into a full-featured environmental monitoring cyberinfrastructure.

In March 2008, I expanded the GBC station by deploying a wireless modem, a single board computer (SBC), and a small pilot WSN consisting of 8 motes, to implement the integrated system shown in Fig. 2.2 in Chapter 2. All the 8 nodes were located around the weather station within one-hop range. At that time, I have yet completely implemented the protocol stack described in Chapter 3, but utilized a simple single-hop transmission protocol. The deployment mainly focused on system integration; it allowed us to extensively evaluate all the composing modules while avoiding the added complexity of multi-hopping. The original system design and initial results were published in [25].

One year later, in March 2009, I relocated the previous 8 nodes and expanded the WSN to a deployment consisting of 16 motes (in two sets of eight motes each) along a cross-sectional transect. This network topology provides an opportunity to collect a duplicated set of soil moisture variation along a cross sectional transect from the river bank (higher elevation and sandy soil) to the weather station (lower elevation and clay soil). Characterizing soil moisture variation with respect to elevation and soil type is vital to understanding vegetation distribution along the floodplain as well as responses to flooding. The new WSN employed the first version of the proposed multihop networking protocols. However, there were still some time synchronization issues which considerably affected both the system reliability and energy efficiency [26].

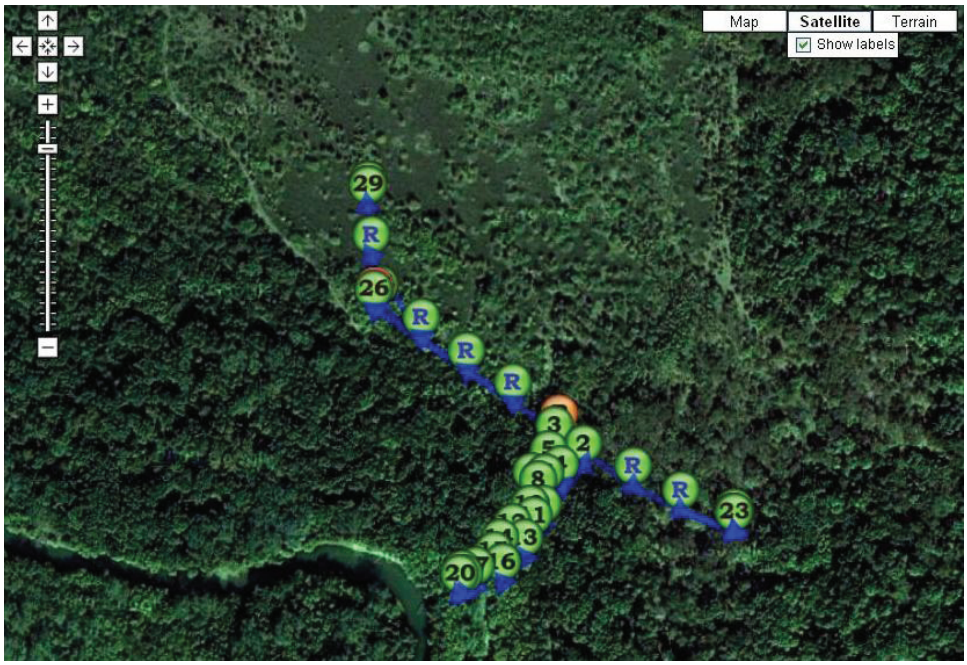

Fig. 5.2. Wireless sensor network topology in GBC site.

Later that same year, four soil sites are identified with different vegetation and soil types based on the soil water dynamic that may have ecological significance, with the collaboration of Department of Environmental Science, University of North Texas (UNT) and city of Denton. I once again expanded the WSN by deploying 3 nodes at each soil site. In addition to the 4 soil sites and the exiting transect, I also added a few routing nodes to link among these nodes. In total there are 35 WSN nodes in the GBC area including 1 BS node, 6 relay nodes and 28 sensor nodes. Current network topology and near-real-time data are available at [56]. A snapshot of the topology is displayed in Fig. 5.2. Routing paths are represented by the blue lines. Sensors are not deployed at regular grid points, mainly because of the irregular layout of trails, trenches, trees, dense bushes, etc. in the field. In December 2009, I released a stable version of the protocol stack which solved the synchronization problem, improved the parent selection algorithm and introduced load-balancing mechanism. However, the system has been frequently offline due to breakdown of the base station. The longest non-interrupted operation for WSN in GBC lasted for a month.

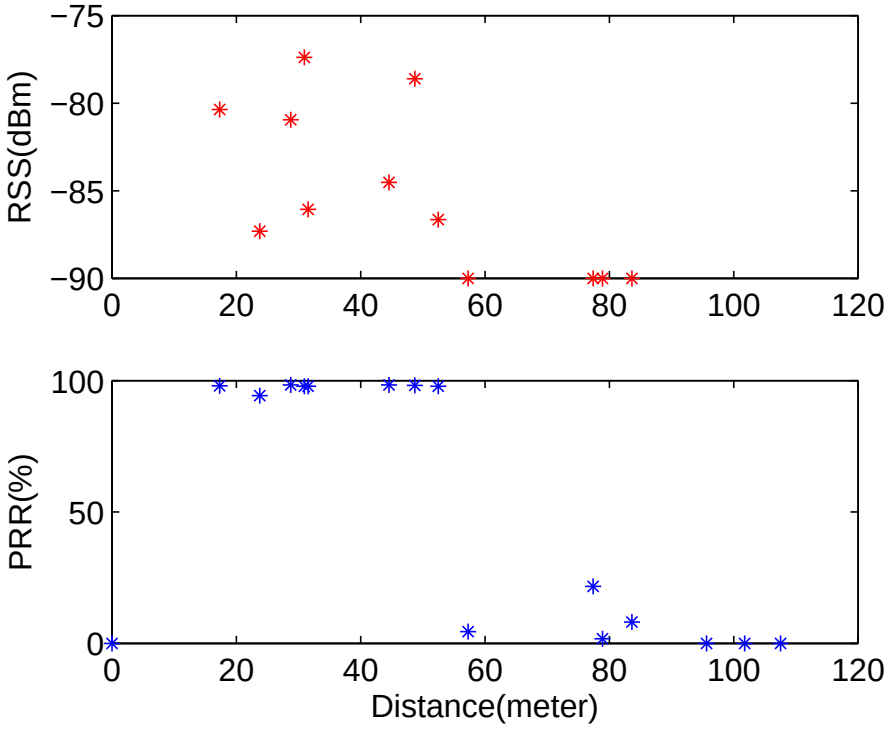

Fig. 5.3. Signal quality measurements over distance.

Prior to the second deployment, I conducted a site survey to measure the one-hop radio communication range between motes in the field that features densely populated trees and grasses. Radio propagation characteristics of the environment vary significantly over time due to seasonal variation of the vegetation in the area. From the measurement results collected in the summer, as shown in Fig. 5.3, it is observed that with a maximal transmission power of 3 dBm, IRIS motes are able to transmit on average 50 m with 95% packet reception rate (PRR) and -88dBm received signal strength (RSS). Thus, motes are deployed with a maximum one-hop distance of about 50 m.

In addition to the GBC site, I have deployed a second weather station with a large-scale WSN in Pecan Creek Waste Water Treatment Plant (PCWWTP), Denton, TX. In contrast to GBC area, the site features mostly grassland with a few bushes. This environment enables us to evaluate the WSN with a more regular topology. As with GBC station, the PCWWTP site has been part of the environmental cyberinfrastructure as well. One can

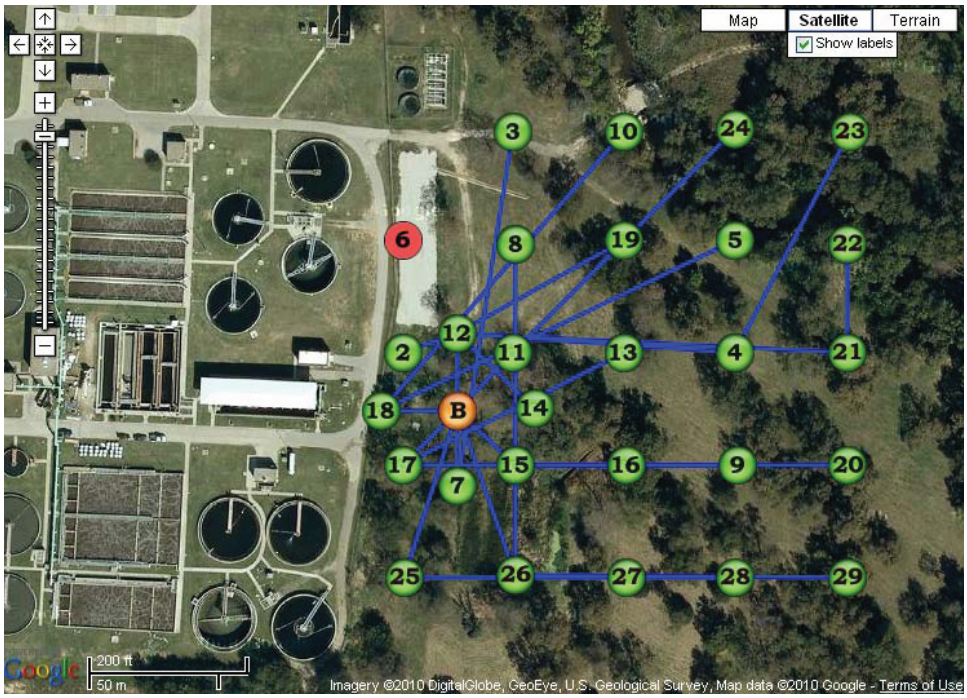

Fig. 5.4. Wireless sensor network topology in PCWWTP site.

find its current network topology and near-real-time data at [56]. As shown in Fig. 5.4, the wireless sensor nodes are arranged in a grid pattern with spacing interval of 45 meters. An inner cycle of 8 sensor nodes were deployed surrounding the base station, in order to balance the data relaying load. The operation in PCWWTP is more successful thanks to the availability of AC power at the base station. Only one node died in the second month of deployment while other nodes had worked for almost 3 months.

5.3. Environmental Data Analysis

Fig. 5.5 shows a subset of data collected by the three motes at the soil site near the Trinity River in GBC area. The three nodes, namely #18, #19, and #20, have been sampling water content every 10 minute, at soil depths of 6", 15" and 30", respectively. Temperature and humidity readings measured by the onboard MDA300 sensors have also been collected together with the soil moisture data. The daily variation of weather condition was well observed by the sensors on motes. We can clearly observe the negative correlation between temperature and humidity. The soil moisture variation along the depth dimension can be

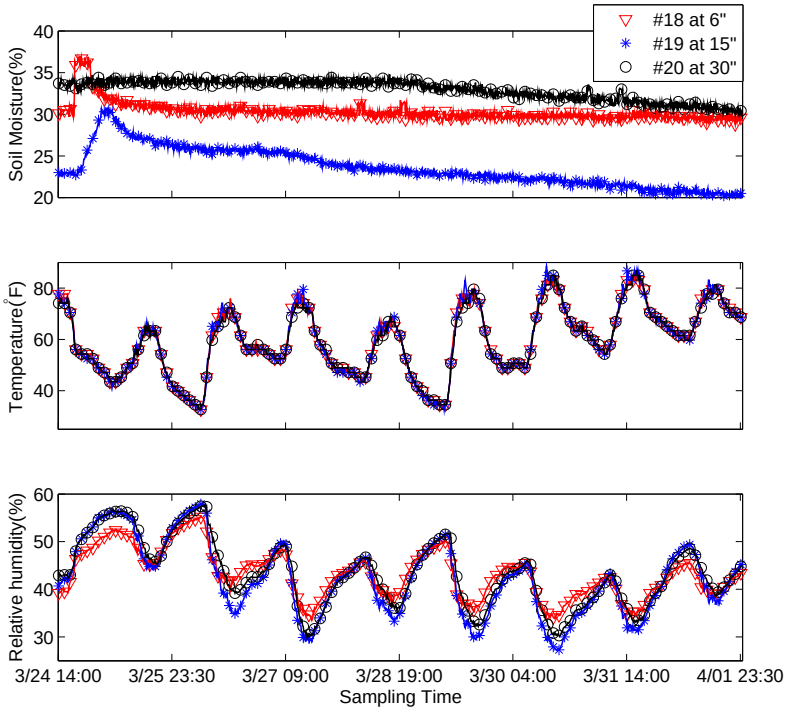

Fig. 5.5. Measurement data collected in soil site I.

characterized from the figure. In general the temporal change of soil moisture becomes less significant as the depth increases. The data segment starts from March 24, 2010 to April 1, 2010, compromising 3612 data points from all three sensor nodes. Among the measurement samples, 53 data points were lost, resulting in 98.53% data reception percentage. Considering that the soil sites are located at the very end of the network, usually more than three hops away from the BS, the data receive rate is reasonable and in line with expectation. A more detailed reliability analysis is given in next section. The missing data in the figure have been interpolated from nearby values.

On the late afternoon of March 24, 2010, the GBC area experienced light rainfall and a sharp drop of temperature, as shown in Fig. 5.5. Fig. 5.6 zooms in the soil moisture curves around the raining event and also displays the precipitation and solar radiation data captured by the datalogger at the same period of time. As the weather began to overcast, the solar radiation reading started to decline, followed by a jump of precipitation monitored

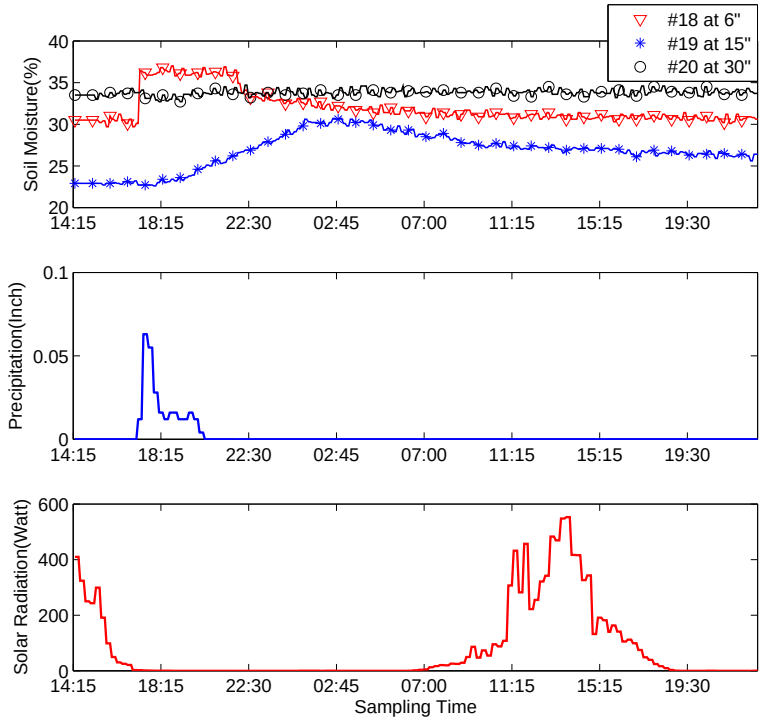

Fig. 5.6. Correlation of soil moisture and precipitation.

by the rain gauge. The surface soil moisture (at 6") reacted first and bounced up by nearly 7%. As the water percolated down through the soil, the soil moisture reading at 15" began to rise. The surface soil water volume peaked as soon as the rain started, and maintained at high level during the rain, but started to drop a few hours after the rain. On the contrary, the middle layer soil moisture responded rather slowly to the event, not peaked until almost ten hours after the rain. Probably due to the due to the small amounts, the soil moisture at deepest layer (30") was not affected by the rain at all. The next day was sunny, as indicated by the solar radiation readings. As the land began to dry out, the soil moisture readings at all levels had been gradually decreasing.

Fig. 5.7 demonstrates the responses of surface soil moisture at different locations to the rain event. In particular, the #21 sensor node is located at soil site IV, which was completely flooded and became a swamp. The soil water readings thus were always high and non-responsive to the precipitation. Sensor #3, #10, #11 are deployed in lowland and

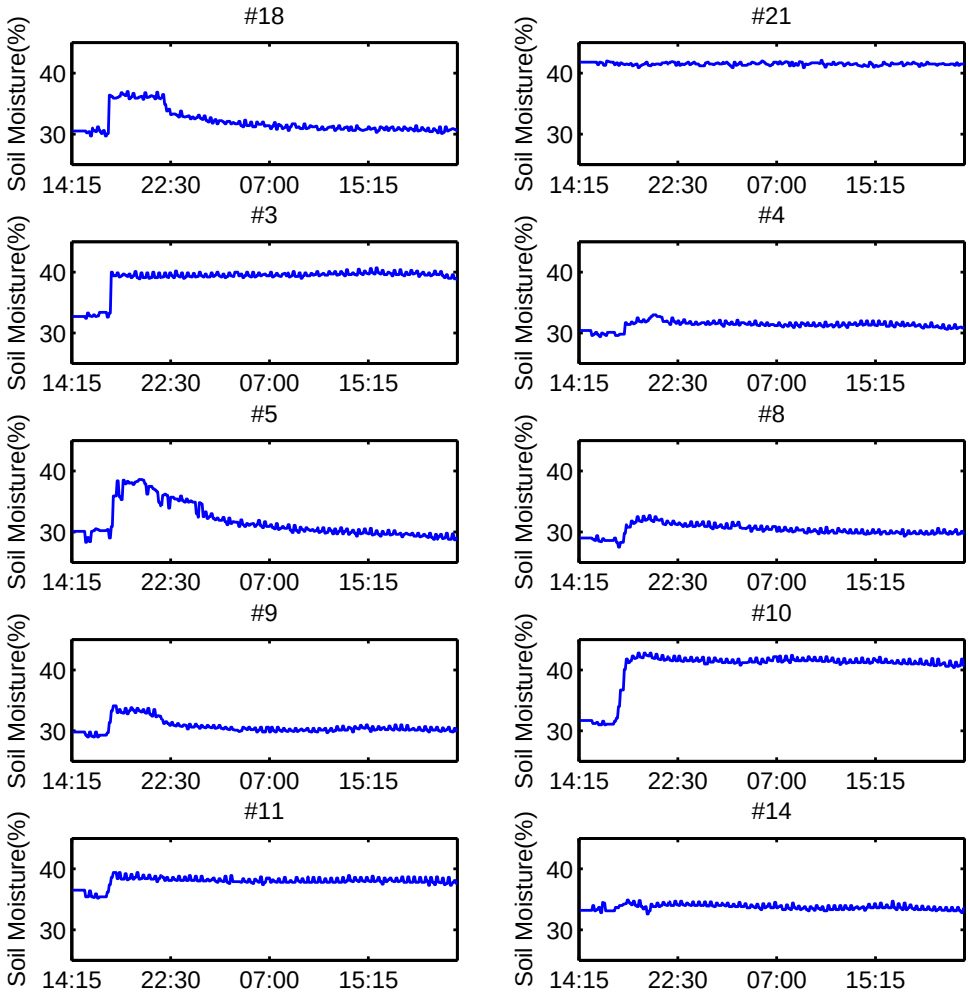

Fig. 5.7. Spatial response to the rain event.

thus subject to standing water. Soil moisture at these locations hiked in the rain and would stay at high level for long time. The remaining sensors are located in places with dense vegetation, i.e. under big trees or inside bushes. Hence smaller increases of soil moisture are observed. The different shapes of soil moisture curves reveal spatial variation characteristics of the soil moisture condition in that area, which is an invaluable input to the hydrologic modeling research.

Table 5.1
Statistics of Sensor Network Status Data

Node ID	Distance to BS (ft)	Hop count	Duty cycle (%)	Receive rate (%)	Delay (sec)
2	103	1	1.03	99.39	15.65
4	126	1.06	1.36	99.47	21.84
32	141	1.16	2.15	99.16	16.86
8	179	1.99	0.85	99.62	31.17
10	243	2.04	0.65	99.47	31.6
33	280	2.04	1.46	99.25	35.25
12	299	2.07	0.78	99.32	28.37
15	386	2.05	1.09	99.54	32.77
34	423	2.51	1.17	99.24	36.06
18	471	2.08	0.9	99.54	31.81
26	565	2.93	1.29	98.87	36.86
35	698	3.45	1.14	98.68	46.51
29	823	3.78	1.78	99.06	49.67

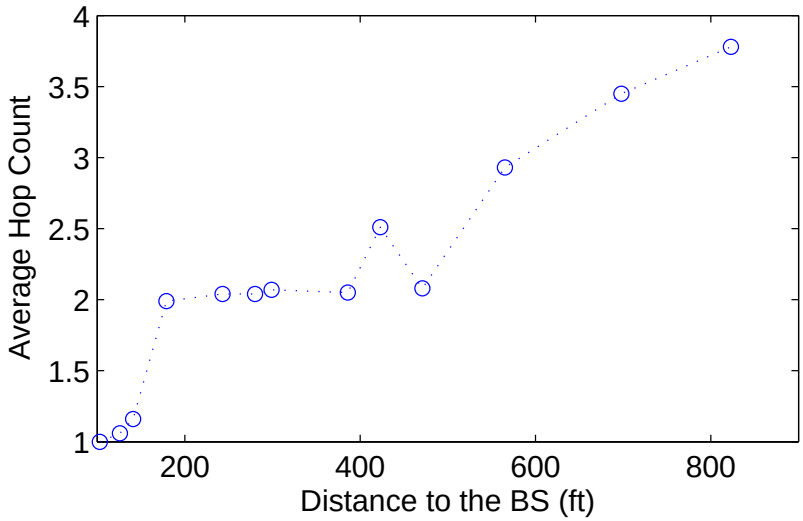

Fig. 5.8. Average hop count versus distance to the BS.

5.4. Network Performance Analysis

Table 5.1 shows a few statistics of a subset of sensors' status data collected from field tests during the month long stable operation in GBC station. The distance between each node and BS is determined using global positioning system (GPS) coordinate measurements. The hop count measurements are the averaged values during the test period as the sensor

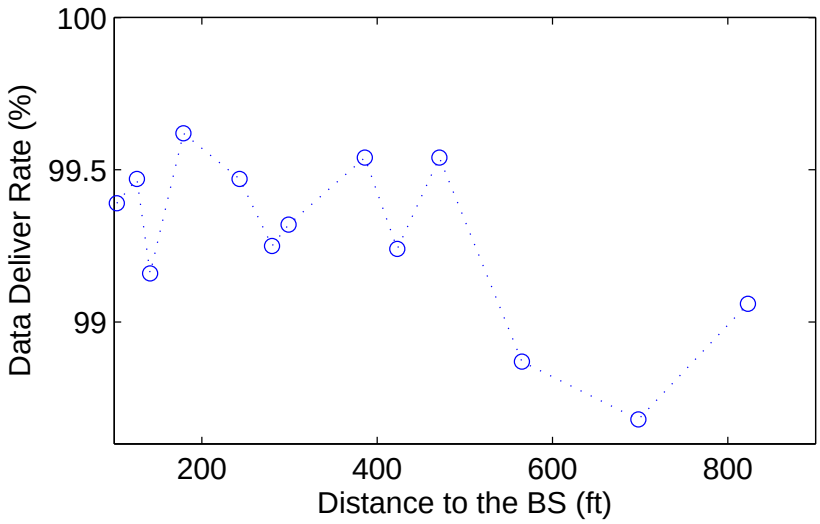

Fig. 5.9. Reliability of data delivery.

network is able to reorganize autonomously in the face of environmental and network changes. From the hop count measurements (between each node and BS) we can clearly observe the tendency of higher hop counts for the nodes with larger distance to the BS, as shown in Fig. 5.8. The reception rate shown in the table is the percentage of the data that are successfully received by BS from each sensor node while each node originates one data sample in every 10 min. This result can also be viewed in Fig. 5.9. From the results, we can observe that though the data deliver rate tends to drop by a small amount, the network achieves close to 99% deliver rate for every sensor.

In the current implementation, each frame of 20s consists of one contention slot of 40 ms and 399 TDMA slots of 40 ms each. With the duty cycle scheduling algorithm, motes are only active during a few TDMA slots to report and relay sensor data and during the contention slot to synchronize time, manage neighbor list, and update parent information. Other than these active periods, motes remain in the sleep mode and consume much less power than in the active mode. As shown in Table 5.1, the average duty cycle of each node is roughly around 1% except for a few dedicated routing nodes with higher duty cycle such as the node #32. The routing nodes are used for extending network coverage and taking higher

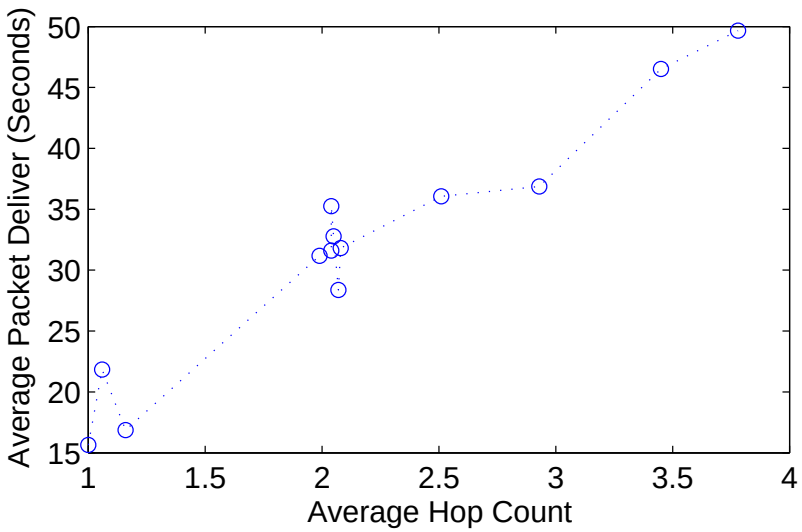

Fig. 5.10. End-to-end data delivery delay versus hop count.

communication loads than normal sensor nodes. The duty cycle determines the average power consumption, as demonstrated in next section.

The last column shows the average end-to-end delay for the data delivery. The data indicates that the delay increases linearly with the hop count, as depicted in the scatter plot Fig. 5.10. Current version of the slot scheduling protocol assigns TDMA slots randomly rather than optimizing for minimum delay. Minimal delay scheduling can be formulated as a graph-theoretical problem [35] and in general it is difficult to solve without introducing significant overhead. Nevertheless, less than ten minutes delay can be tolerated in the application, as indicated in Section 3.1.

5.5. Energy Consumption Analysis

From experimental measurements, a bare IRIS mote draws around 18 mA in the active mode. However, in order to examine sensor status in the field, two light-emitting diodes (LEDs) are employed to indicate packet transmission and duty cycling, adding 6 mA current draw in the active mode. In the sleep mode, the radio transceiver is disabled and the central processing unit (CPU) wakes up occasionally to handle hardware interrupt routines and software events, such as timer services and counter updates, in order to maintain network stack and remain synchronized. With all peripheral devices except the hardware clock turned

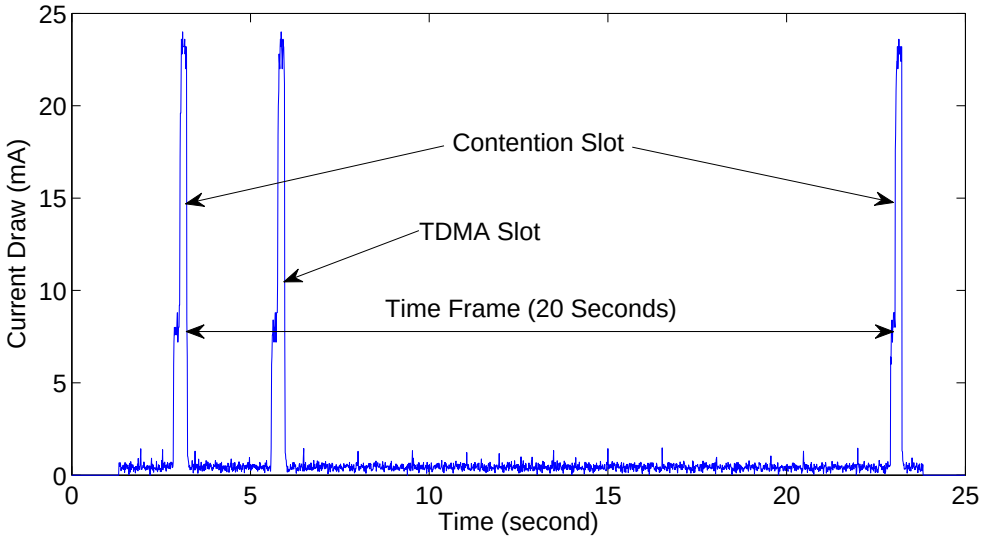

Fig. 5.11. Measurement of the current draw and duty cycle of a mote.

off a minimum of 70 μA in sleep mode can be achieved. The use of the MDA300 data acquisition board introduces extra 0.45 mA current consumption, in both active and sleep modes. Consequently, as shown in Fig. 5.11, a sensor node consumes 24 mA when working actively in contention and TDMA slots, and about 0.6 mA power when sleeping rest of the time. The average current draw of a sensor node is about 0.84 mA with 1% duty cycle. Thus, with two fresh 2200 mAh AA-size batteries, a sensor node can sustain for about 3 months.

Fig. 5.12 reveals the energy consumption by showing battery measurements from 4 nodes in PCWWTP site. The system was deployed on December 11, 2009. Sensors were powered by two freshly-charged 2200mAh nickel-metal hydride (NiMH) batteries. The system was operational for nearly 3 months until most sensor died in early March, 2010. A few sensors including #3 still had remaining power functioning until March 16, 2010.

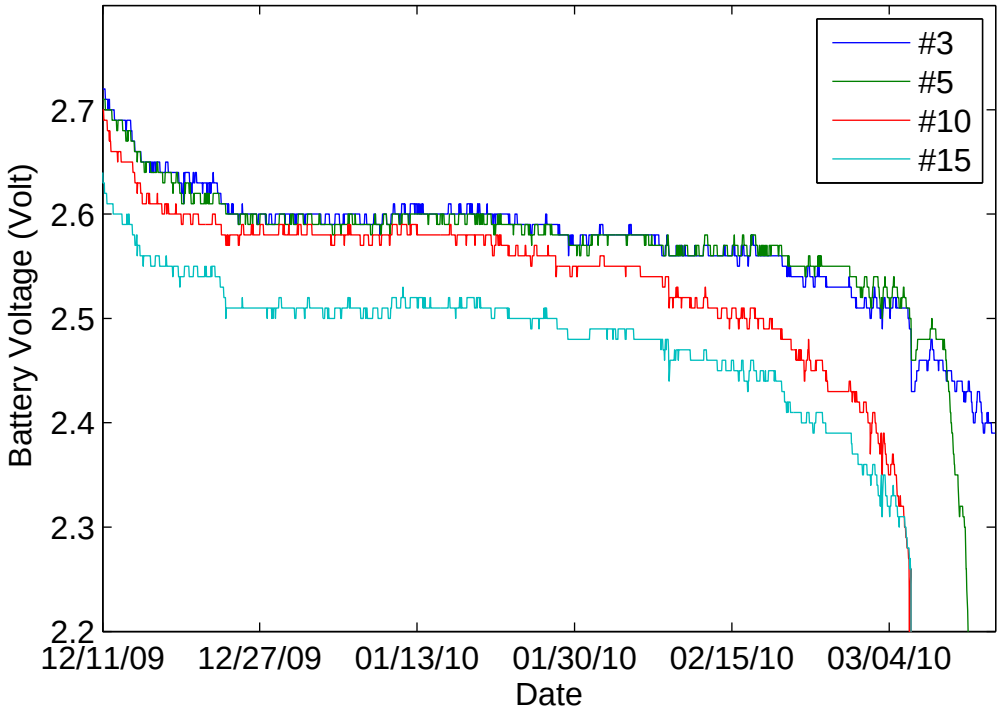

Fig. 5.12. Battery measurements in Pecan Creek site.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Environmental monitoring represents a major application domain for wireless sensor networks (WSN). However, despite significant advances in recent years, there are still many challenging issues to be addressed to exploit the full potential of the emerging WSN technology. This research strives to solve the challenges faced in many WSN researches and make significant practical contributions to WSN community. Specifically, this dissertation explores the design, implementation and deployment of a suite of software, protocols and algorithms for energy-efficient multi-hop wireless networks with near-real-time delay constraints. The proposed solution will be very useful for all low data rate monitoring applications including environmental monitoring.

Two wireless sensor networks have been deployed in remote field stations to monitor soil moisture along with other environmental parameters. Performance of the networks built upon the proposed solution meets the design requirements prescribed for the soil moisture monitoring application. As parts of the ever-growing environmental monitoring cyberinfrastructure, these networks have been integrated into the Texas Environmental Observatory (TEO) system for long-term operation. Future works include implementing end-to-end acknowledgement and end-to-end retransmission mechanisms to further increase its reliability, and optimizing slot allocation to reduce delay with minimum overhead.

As network size scales up, the duty-cycle will be inevitably boosted in order to obtain sufficient throughput for accommodating increasing amount of data packets. To address the issue, I propose a novel error-bounded adaptive sampling algorithm which brings down not only the communication cost but also the sensing overhead. Reducing the data originated from sensor nodes reduces not only the energy consumption of source nodes, but also the

processing and communication loads at all intermediate nodes in multihop networks. I am currently in the process of algorithm evaluation and simulation.

BIBLIOGRAPHY

[1] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks", *IEEE Computer*, pp. 41–49, Aug. 2004.

[2] "MoteLab: Harvard Sensor Network Testbed", http://motelab.eecs.harvard.edu.

[3] "ExScal: Extreme Scale Wireless Sensor Networking", http://cast.cse.ohio-state.edu/exscal.

[4] "Kansei: Sensor Testbed for At-Scale Experiments", http://ceti.cse.ohio-state.edu/kansei.

[5] "CitySense: An Open, Urban-Scale Sensor Network Testbed", http://www.citysense.net.

[6] "SCADDS: Scalable Coordination Architectures for Deeply Distributed Systems", http://www.isi.edu/scadds/testbeds.

[7] S. Cheng and R. Wang, "An approach for evaluating the hydrological effects of urbanization and its application", *Hydrological Processes*, 16:1403–1418, 2002

[8] Open Geographic Consortium Inc., "Sensor Web Enablement WG.", http://www.opengeospatial.org.

[9] B. Harrington, Y. Huang, J. Yang, and X. Li, "Energy efficient map interpolation for sensor fields using Kriging", *IEEE Trans. on Mobile Computing*, accepted for publication, in press.

[10] W. Stallings, *Wireless Communications & Networks, 2nd Edition*, Prentice Hall, 2004.

[11] Technologic Systems Inc., "ARM Single Board Computers for Embedded Systems", http://www.embeddedarm.com.

[12] NovaLynx Corporation, "Power Budget Calculations", http://www.novalynx.com.

[13] UAH VAST, "SensorML", http://vast.uah.edu/SensorML.

[14] Google Inc., "Keyhole Markup Language", http://code.google.com/apis/kml.

[15] 52north.org, "52North Sensor Web", http://52north.org.

[16] Google Inc., "Google Maps API", http://code.google.com/apis/maps.

[17] Sun Microsystems Inc., "Metro Web Services", http://java.sun.com/webservices.

[18] Wikipedia.org, "Faceted search", http://en.wikipedia.org/wiki/Faceted_browser.

[19] F. C. Delicato, P. F. Pires, L. Pinnez, L. Fernando, and L. F. R. da Costa, "A flexible web service based architecture for wireless sensor networks", *Proc. of 23rd Int'l Conf. on Distributed Computing Systems Workshops*, May. 2003.

[20] J. Zhang, Q. Hart, M. Gertz, C. Rueda, and J. Bergamini, "Sensor data dissemination systems using Web-based standards: a case study of publishing data in support of evapotranspiration models in California", *Civil Engineering and Environmental Systems*, vol. 26, issue 1, pp. 35-52, Mar. 2009.

[21] R. M. Garcia, P. Carvalhal, M. J. Ferreira, L. F. Silva, H. Almeida, C. Santos, and J. A. Afonso, "A Flexible framework for data exchange and presentation between wireless sensor networks and personal devices", *The Int'l Conf. on "Computer as a Tool"*, Sep. 2009.

[22] A. Sleman and R. Moeller, "Integration of wireless sensor network services into other home and industrial networks using device profile for web services (DPWS)", *The 3rd Int'l Conf. on Information and Communication Technologies: From Theory to Applications (ICTTA)*, Apr. 2008.

[23] Y. Kawahara, N. Kawanishi, M. Ozawa, H. Morikawa, and T. Asami, "Designing a framework for scalable coordination of wireless sensor networks, context information and web services", *The 27th Int'l Conf. on Distributed Computing Systems Workshops (ICDCSW)*, Jun. 2007.

[24] Feng Zhao, Leonidas Guibas, *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2004

[25] Jue Yang, Chengyang Zhang, Xinrong Li, Yan Huang, Shengli Fu, and Miguel Acevedo, "An environmental monitoring system with integrated wired and wireless sensors", *International Conference on Wireless Algorithms, Systems and Applications (WASA)*, Dallas, TX, Oct. 2008.

[26] Jue Yang, Chengyang Zhang, Xinrong Li, Yan Huang, Shengli Fu, and Miguel Acevedo, "Integration of wireless sensor networks in environmental monitoring cyber infrastructure", *Wireless Networks*, Springer/ACM, DOI: 10.1007/s11276-009-0190-1, Jun. 2009.

[27] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring", *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 88–97, New York, NY, USA, ACM, 2002.

[28] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks", *IEEE Computer*, pp. 50–56, Aug. 2004.

[29] R. Musaloiu-E, A. Terzis, K. Szlavecz, A. Szalay, J. Cogan, and J.Gray, "Life under your feet: A wireless soil ecology sensor network", *Proc. of the Third Workshop on Embedded Networked Sensors (EmNets)*, May. 2006.

[30] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer, "A reactive soil moisture sensor network: Design and field evaluation", *Int'l Journal of Distributed Sensor Networks*, pp. 149-162, vol. 1, no. 2, Apr.-Jun. 2005.

[31] Igor Talzi, Andreas Hasler, Stephan Gruber and Christian Tschudin, "PermaSense: Investigating Permafrost with a WSN in the Swiss Alps", *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets'07)*, pp. 8–12, Jun. 2007.

[32] Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M., et al., "SensorScope: Out-of-the-Box Environmental Monitoring", *The 7th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pp. 332–343, 2008.

[33] De Roure, D., "Floodnet: a new flood warning system", *Royal Academy of Engineering Quarterly*, 23, pp. 48-51, 2005.

[34] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network", *Proc. Second European Workshop on Wireless Sensor Networks (EWSN'05)*, Jan. 2005.

[35] Rowe A, Mangharam R, Rajkumar R, "RT-Link: a time-synchronized link protocol for energy-constrained multi-hop wireless networks", *Third IEEE international conference on sensors, mesh and ad hoc communications and networks (IEEE SECON)*, 2006.

[36] W. Ye, J. Heidemann, D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks", *Proc. 21st Int. Annu. Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, vol.3, pp. 1567–1576, Jun. 2002.

[37] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks", *Proc. of the Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, 2003.

[38] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks", *Proc. of the Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, 2004.

[39] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: A hybrid MAC for wireless sensor networks", *Proc. of the Int'l Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[40] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks", *Proc. of the Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, 2003.

[41] L.F.W. van Hoesel and P.J.M. Havinga, "A lightweight medium access protocol for wireless sensor networks", *1st International Conference on Networked Sensing Systems*, 2004.

[42] Hari Balakrishnan et al., "The distance-2 matching problem and its relationship to the mac-layer capacity of ad hoc wireless networks", *IEEE Journal on Selected Areas in Comm.*, 22(6):1069–1079, Aug. 2004.

[43] G. Lu, N. Sandagopan, B. Krishnamachari, and A. Goel, "Delay Efficient Sleep Scheduling in Wireless Sensor Networks", *IEEE Infocom 2005*, Mar. 2005.

[44] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol", *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys)*, Nov. 2004.

[45] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts", *Proc. of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Dec. 2002.

[46] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks", *Proc. Int. Conf. Embedded Networked Sensor Systems (SenSys)*, Nov. 2003.

[47] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks", *Proc. Int. Conf. Embedded Networked Sensor Systems (SenSys)*, Nov. 2003.

[48] Perkins, C., Belding-Royer, E., Das, S., "Ad hoc on-demand distance vector (AODV) routing", *IETF RFC 3561*, Jul. 2003.

[49] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Jan. 2000.

[50] Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., Wicker, S., "Complex behavior at scale: An experimental study of low-power wireless sensor networks", University of California, Los Angeles, Technical Report UCLA/CSD-TR-02-0013, 2002.

[51] Zigbee Alliance, "Zigbee Alliance", http://www.zigbee.org.

[52] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey". *Wireless Communications, IEEE*, 11(6):6-28, Dec. 2004.

[53] Y. Xu, J. Heidemann, D. Estrin, "Geography-informed Energy Conservation for Ad-hoc Routing", *In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 70–84, 2001.

[54] Simon, G. et al. "Sensor Network-Based Countersniper System", *the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2004.

[55] N. Bulusu, J. Heidemann, D. Estrin, "GPS-less low cost outdoor localization for very small devices", Computer Science Department, University of Southern California, Technical report 00-729, Apr. 2000.

[56] Texas Environmental Observatory, "TEO Online", http://www.teo.unt.edu.

[57] University of North Texas, "Environmental Conditions Online of DFW MetroPLEX (ECOPLEX)", http://www.ecoplex.unt.edu.

[58] TinyOS Community Forum, "TinyOS: An open-source OS for the networked sensor regime", http://www.tinyos.net.

[59] Crossbow Technology Inc., "Crossbow Technology : Wireless Sensor Network", http://www.xbow.com.

[60] Ubiquiti Networks Inc. "NanoStation", http://www.ubnt.com.

[61] Decagon Devices Inc., "Decagon Soil Moinsture Systems", http://www.decagon.com/soil_moisture.

[62] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks", *IEEE Communications Magazine*, Aug. 2002.

[63] Milenkovi, A., C. Otto and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation", *Computer Communications*, vol. 29, pp. 2521-2533, 2006.

[64] Zhuang, L.Q., Liu, W., Zhang, J.B., Zhang, D.H., Kamajaya, I., "Distributed asset tracking using wireless sensor network", *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp.1165-1168, Sep. 2008.

[65] K. Gill, S.-H. Yang, F. Yao, and X. Lu, "A ZigBee-based home automation system", *IEEE Trans. on Consumer Electronics*, vol. 55, no. 2, pp. 422-430, May. 2009.

[66] S.-H. Hong, B. Kim, and D.-S. Eom, "A base-station centric data gathering routing protocol in sensor networks useful in home automation applications", *IEEE Trans. on Consumer Electronics*, vol. 53, no. 3, pp. 945-951, Aug. 2007.

[67] V.C. Gungor and G.P. Hancke, "Industrial wireless sensor networks: challenges, design principles, and technical approaches", *IEEE Trans. on Industrial Electronics*, vol. 56, no. 10, pp. 4258-4265, Oct. 2009.

[68] K.A. Agha, M.-H. Bertin, T. Dang, A. Guitton, P. Minet, T. Val, and J.-B. Viollet, "Which wireless technology for industrial wireless sensor networks? The development of OCARI technology", *IEEE Trans. on Industrial Electronics*, vol. 56, no. 10, pp. 4266-4278, Oct. 2009.

[69] B. Lu and V.C. Gungor, "Online and Remote Motor Energy Monitoring and Fault Diagnostics Using Wireless Sensor Networks", *IEEE Trans. on Industrial Electronics*, vol. 56, no. 10, pp. 4651-4659, Oct. 2009.

[70] D.-S. Kim, S.-Y. Lee, K.-H. Won, D.-J. Chung, and J.-H. Kim, "Time-synchronized forwarding protocol for remote control of home appliances based on wireless sensor network", *IEEE Trans. on Consumer Electronics*, vol. 53, no. 4, pp. 1427-1433, Nov. 2007.

[71] S. Raje and Q. Liang, "Time synchronization in network-centric sensor networks", *Proc. of IEEE Radio and Wireless Symposium*, 2007.

[72] K. E. Atkinson, *An introduction to Numerical Analysis, 2nd edition*, John Wiley & Sons, Inc., 1988.

[73] Atmel Corporation, "AT86RF230, Rev. E", http://www.atmel.com.

[74] Atmel Corporation, "ATmega640/1280/1281/2560/2561 Preliminary, Rev. L", http://www.atmel.com.

[75] J. Han and M Kamber, *Data Mining: Concepts and Techniques, 2nd edition*, Elsevier Inc., 2006.

[76] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall PTR, 1993.

[77] X. Li, "An iterative NLOS mitigation algorithm for location estimation in sensor networks", *Proc. of The 15th IST Mobile & Wireless Communications Summit*, Jun. 2006.

[78] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts, 7th edition*, John Wiley & Sons, Inc., 2005.

[79] Symmetricom Inc., "Stochastic model estimation of network time variance", White Paper, 2003, http://www.ntp-systems.com.