IMPROVEMENT OF A THREE-TIER WIRELESS SENSOR NETWORK FOR ENVIRONMENT MONITORING

by

XU WANG

B. Eng., China Agricultural University, P. R. China, 2006
M. Eng., China Agricultural University, P. R. China, 2009

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Biological and Agricultural Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Abstract

A three-tier wireless sensor network (WSN) was developed and deployed to remotely monitor suspended sediment concentration and stream velocity in real-time. Two years of field experiments have demonstrated the achievement of such capabilities. But several weak points emerged and required essential performance improvement and additional research on the radio propagation mechanism within the original three-tier WSN.

In the original three-tier WSN, long time delay, potential data loss, and limited network throughput all restricted the network transmission performance. Upon the above issues, the transmission delay was reduced through shortening the raw data storage buffer and the data packet length; the data loss rate was decreased by adopting a mechanism using semaphores and adding feedback after data transmission; the network throughput was enlarged through the event- and time-driven scheduling method.

In order to find a long-range wireless transmission method as an alternative to the commercial cellular service used in the original WSN, a central station using meteor burst communication (MBC) technology was developed and deployed. During an 8-month field test, it was capable of performing long distance communication with a low data loss rate and transmission error rate. But due to unstable availability of the meteor trails, the MBC network throughput was constrained.

To reduce in-situ maintenance, over-the-air programming was implemented. Thus, programs running in the central station and the gateway station can be updated remotely.

To investigate the radio propagation in densely vegetative areas, a 2.4 GHz radio propagation path loss model was derived to predict the short-range path loss from the path loss in the open area and the path loss due to dense vegetation. In addition, field experiments demonstrated that ambient air temperature, relative humidity, and heavy rainfall could also affect wireless signal strength.

IMPROVEMENT OF A THREE-TIER WIRELESS SENSOR NETWORK FOR ENVIRONMENT MONITORING

by

XU WANG

B. Eng., China Agricultural University, P. R. China, 2006
M. Eng., China Agricultural University, P. R. China, 2009

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Biological and Agricultural Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Naiqian Zhang

Copyright

Abstract

A three-tier wireless sensor network (WSN) was developed and deployed to remotely monitor suspended sediment concentration and stream velocity in real-time. Two years of field experiments have demonstrated the achievement of such capabilities. But several weak points emerged and required essential performance improvement and additional research on the radio propagation mechanism within the original three-tier WSN.

In the original three-tier WSN, long time delay, potential data loss, and limited network throughput all restricted the network transmission performance. Upon the above issues, the transmission delay was reduced through shortening the raw data storage buffer and the data packet length; the data loss rate was decreased by adopting a mechanism using semaphores and adding feedback after data transmission; the network throughput was enlarged through the event- and time-driven scheduling method.

In order to find a long-range wireless transmission method as an alternative to the commercial cellular service used in the original WSN, a central station using meteor burst communication (MBC) technology was developed and deployed. During an 8-month field test, it was capable of performing long distance communication with a low data loss rate and transmission error rate. But due to unstable availability of the meteor trails, the MBC network throughput was constrained.

To reduce in-situ maintenance, over-the-air programming was implemented. Thus, programs running in the central station and the gateway station can be updated remotely.

To investigate the radio propagation in densely vegetative areas, a 2.4 GHz radio propagation path loss model was derived to predict the short-range path loss from the path loss in the open area and the path loss due to dense vegetation. In addition, field experiments demonstrated that ambient air temperature, relative humidity, and heavy rainfall could also affect wireless signal strength.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all I would like to express my deepest gratitude to my major professor, Dr. Naiqian Zhang, for his academic guidance, supervision, and financial support throughout my Ph.D. study. His wisdom, knowledge and commitment to the highest standards inspired and motivated me. His diligence and seriousness in research, and his independent thinking and open-minded attitude towards novelties will serve as a model for my future academic life.

It is a pleasure to extend my gratitude to my supervisory committee: Dr. Mitch Neilsen, Dr. Balasubramaniam Natarajan, and Dr. Ning Wang. Thanks to Dr. Neilsen for his kind assistance in real-time system study. Thanks to Dr. Natarajan for his valuable suggestions on wireless communication. Thanks to Dr. Wang for spending her precious time on guiding and helping the entire research project. I also thank Dr. Michael O'Shea for willing to serve as my outside chairperson. Deep gratitude is extended to Dr. Joseph Harner, Head of the Biological and Agricultural Engineering Department, for his support and encouragement.

My profound appreciation and gratitude are given to my beloved parents and fiancée, for their encouragement, support, patience, and unwavering love throughout my academic career, to whom I owe the most sincere thanks.

I am grateful for assistances and suggestions from many experts. Thanks to Mr. Darrell Oard for his advice and assistance in wireless sensor network field deployment. Thanks to Mr. Brandon Lee Lantz for helping me maintain the wireless devices in Fort Riley, KS. Thanks to Mr. Carl Johnson for his great effort in field experiments.

I would like to express special thanks to Dr. Wei Han, Dr. Joseph Dvorak, Mr. Daniel Bigham, Mr. Marvin Petingco, and Mr. Peyman Taher, for their assistance as my colleagues during the ESTCP project. I would also like to thank Dr. Peng Li and his wife Ms. Ning Tang for their help during my Ph. D study.

I am heartily thankful to the department staff, Ms. Barbara Moore, Mr. Randy Erickson, Ms. Cindy Casper, Ms. Arlene Jacobson and Ms. Lou Ann Claassen for helping me on all aspects of my study and research in the BAE department.

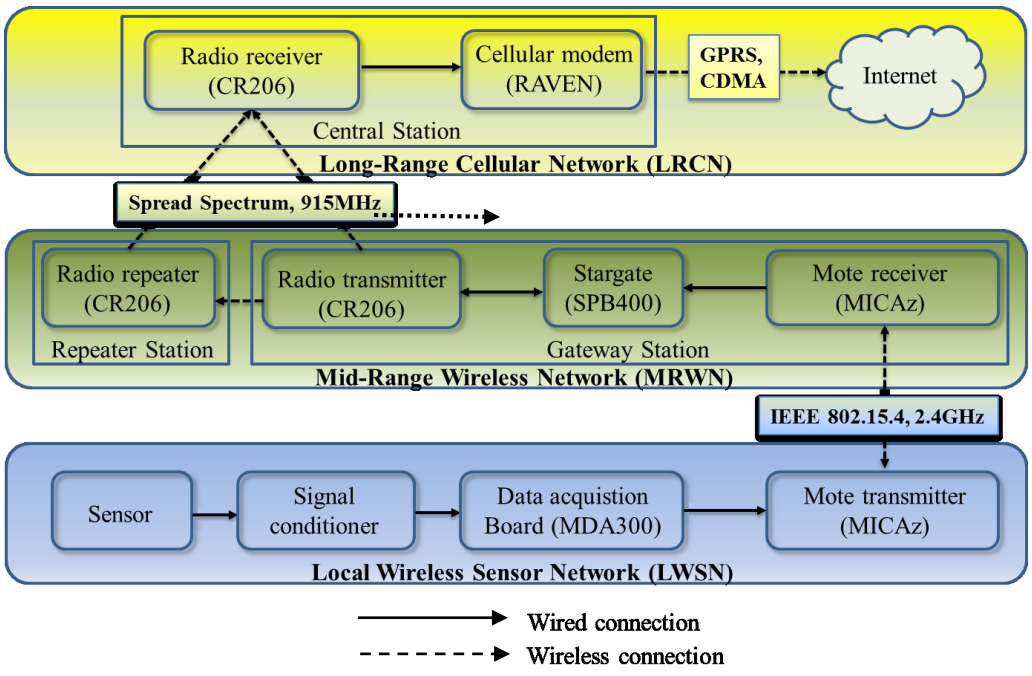
Lastly, I offer my regards and blessings to all of those who supported me in any respect during my life in Kansas State University.

Chapter 1 - INTRODUCTION AND OBJECTIVES

Streams and creeks on military installations often experience sediment loads and reduced water quality due to erosion from unimproved tactical trails, increased training activities, and new construction. Current standard practices to estimate suspended sediment concentrations (SSC) have been reported as unreliable, labor intensive, and expensive. Therefore, a project was proposed to establish a methodology to monitor the SSC and quantify the sediment flux through measurement. Under this project, an SSC sensor and a three-tier wireless sensor network (WSN) were demonstrated and validated for continuous, in-situ, real-time measurement and web-based, installation-scale, remote monitoring of SSC and flow velocity.

Since 2008, a three-tier WSN has been developed and deployed at three military installations - Fort Riley in Kansas, Fort Benning in Georgia, and Aberdeen Proving Ground in Maryland. The three-tier WSN included a local wireless sensor network (LWSN), a mid-range wireless network (MRWN) and a long-range cellular network (LRCN) (Han 2011). The basic function of this multi-tier network architecture was to transmit data from multiple, signal-unfriendly areas to a central station in an area with good, commercial cellular coverage. A block diagram of the three-tier WSN is shown in Figure 1.1.

Figure 1.1 Block diagram of the three-tier WSN



The LWSN was in charge of wirelessly transmitting SSC and flow velocity data from sensors to a gateway station. The area covered by the wireless transmission was usually close to a stream or a bridge, and surrounded by dense vegetation. Due to signal reflection, diffraction, and scattering, wireless propagation was hampered by the vegetation in general. In addition, commercial cellular network services were weak or non-existent in these areas. In order to collect raw SSC data from multiple sensors, short range (within 100 meters) wireless devices were deployed to transmit data from distributed sensor nodes to a gateway station. The wireless transmission within this tier was achieved by wireless Motes (MICAz, MEMSIC Inc., Andover, MA), including multiple transmitters attached to the sensor control boards, and one receiver attached to a Stargate (SPB400, MEMSIC Inc., Andover, MA). The wireless communication was based on the ZigBee (IEEE 802.15.4, 2.4G Hz) protocol.

The MRWN relayed the data, through a moderately long distance (up to 8 kilometers from the experience in this project), from the signal-unfriendly gateway stations to a location with satisfactory commercial cellular coverage. Data packets from one or multiple LWSNs were transmitted from gateway stations to a central station. Repeater stations were deployed optionally in this tier to either extend the network coverage or to improve signal strength. The Stargate in the gateway station accessed the raw data from the Mote attached to it. After the data was processed and encapsulated into a data packet, Stargate would output the packet to the datalogger (CR206, Campbell Scientific Inc., Logan, UT) through an RS232 serial port. Then the data packet would be transmitted from the gateway station to the central station, which was achieved by the add-on 915 MHz radio in the CR206 datalogger using Frequency Hopping Spread Spectrum (FHSS) technology.

In the LRCN, a commercial cellular data service was used to further transmit data to the database server through the Internet. The datalogger at the central station output the data packet to a cellular modem (RAVEN, Sierra Wireless, Richmond, BC, Canada) through serial communication. The cellular modem would then send the data via the commercial GPRS or CDMA service. When the server received the data, the SSC/velocity information would be presented on a website in real-time.

Field experiments from 2008 to 2009 have demonstrated that the original three-tier WSN was capable of real-time monitoring of SSC and velocity in streams in remote areas, where no

commercial wireless data service was available. Meanwhile, several problems were found in system performance, indicating the need for modification and improvement.

Performance of a wireless system was usually evaluated on data loss rate, network throughput, and real-time properties. In the original three-tier WSN, the data loss rate, which was defined as the ratio of data packets received per day to data packets generated per day, was found relatively high. The data parsing and packaging methods used in the Stargate program and the data packets scheduling methods used in the datalogger program restricted the network throughput. Moreover, the purely time-driven scheduling methods extended the interval between the generation of the raw data and the final display of the SSC value, and increased the risk of missing the transmission deadline. To solve the above problems, new wireless transmission methods were developed in this study to improve the performance of the existing WSN.

Although most commercial cellular network services have been proven reliable for long distance communication, several unusual conditions would cause interruptions in cellular services and, subsequently, failures in WSNs that use the commercial cellular service as a part of the transmission path. Examples of these conditions included unexpected stoppage from the service provider, weak cellular coverage, natural disasters such as earthquake, and man-made ionospheric disturbance such as nuclear explosion. An alternative to cellular service for long distance communication was meteor burst communication (MBC), which used a medium that was not affected by above mentioned constraints. In this study, MBC as a replacement of the commercial cellular service in the LRCN was also investigated.

As all the wireless transmission devices were installed in remote, unattended areas, maintenance of these devices, including troubleshooting, software upgrading, and system restarting, required field trips of technicians, resulting in great labor and travel costs. If these equipment could be configured or re-programmed remotely, these costs could be reduced enormously. Therefore, a study of over-the-air programming (OAP) was conducted to enable the remote programming function in the existing WSN.

Because all the SSC/velocity sensors were installed in streams in remote areas, the radio transmitters and receivers in the LWSN were generally surrounded by dense vegetation, such as trees and tall grass, which greatly affected the radio propagation. In this study, a radio propagation model in densely vegetated areas was studied. This model may provide guidelines for selections of antennas and transmitter/receiver locations in the future.

Thus, the overall objective of this study was to improve the performance of an existing three-tier WSN.

The specific objectives were:

1) To develop new wireless transmission mechanisms to improve the network performance. The tasks included were

- developing raw data encapsulation, buffering and re-transmission methods to reduce the data transmission loss rate,
- developing data packet scheduling methods to increase the data throughput in the WSN, and
- developing a response-enabled transmission method to shorten the data package transmission time.

2) To deploy and test the MBC technology as an alternative to commercial cellular services for long range wireless transmission. The tasks included were selecting equipment for MBC,

- select the site that best fit the MBC transmission,
- developing the data communication method for MBC, and
- comparing the MBC performance with cellular network services.

3) To develop the OAP functions on the existing three-tier WSN. The tasks included were

- selecting equipment and data transmission service for OAP and
- developing data communication methods.

4) To study the radio propagation patterns in densely vegetated environment. The tasks included were

- developing a path loss model to predict the path loss in point-to-point wireless transmission within the LWSN tier and
- investigating other factors that may impact the signal strength.

Chapter 2 - LITERATURE REVIEW

2.1 Wireless sensor networks for environmental monitoring

A WSN is a system comprised of radio frequency (RF) transceivers, sensors, microcontrollers, and power sources. It is a technology that can provide processed, real-time field data from distributed sensors. Its advantages, including flexible networking, multi-data acquisition, little infrastructure, low energy cost, and low dollar cost, all contribute to the wide application prospect in environment monitoring field (Yick et al. 2008). With different types of sensors, a WSN can measure numerous factors in the environment, such as temperature, atmospheric pressure, and relative air humidity. In the agricultural environment monitoring field, various WSN systems were developed to monitor soil moisture (Wark et al. 2007), soil electrical conductivity (Li et al. 2012), ammonia concentration (Ham et al. 2012), carbon dioxide concentration (Wang et al. 2010), dissolved oxygen concentration (Lawlor et al. 2012), and suspended sediment concentrations (Stewart et al. 2014). The environment WSNs can monitor varies from small scale ones, such as greenhouses (Kim et al. 2012), cattle feedlots (Ham et al. 2012), to medium scale ones, such as vineyards (Burrell et al. 2004) and fish ponds (Zhou et al. 2013), and further to large scale ones, such as farms (Pierce et al. 2008) and streams (Han et al. 2008). In fact, environmental monitoring is probably one of the fields that provide the most suitable scenarios for the applications of WSN.

The swift development of microcontroller and wireless communication technologies has allowed image and video information to be processed much faster with less power, and transmitted via longer distance. A large amount of image and video data have been collected at the wireless sensor nodes and transmitted to the sink nodes for environmental monitoring. Antonio et al. (2011) added video-surveillance function to a crop monitoring WSN to protect the crops and farm equipment from intruders. Xiao et al. (2012) reported a multi-sensor network for forest fire detection, which integrated temperature, humidity, and wind speed sensors with cameras to detect smoke. Liang et al. (2012) developed a WSN to acquire the videos and images of the crop growth situation in real-time with a flexible video resolution based on the bandwidth of the wireless network. To reduce the image and video transmission power and the influence by the peripheral environment, Zhang et al. (2012) conducted a research to develop a method to

compress the crop images by wavelet transformation, and improved the wireless transmission efficiency.

Another trend of WSN for environmental monitoring has emerged with the function not only to support environment sensing but also to change the environment with distributed controllers. Robert et al. (2013) developed a WSN integrated with sensing nodes and operating nodes to perform precision irrigation. The sensing nodes reported the soil moisture information to the Gateway Station, while the operating nodes received the command from the Gateway Station and controlled the relays to actuate latching valves. Costas et al. (2012) investigated a mobile agro-environmental location aware system in ground spray applications against olive fruit flies. The system integrated the built-in spays with WSN, GIS, and also GPS technologies. It reduced the amount of spraying by analyzing the spatiotemporal characteristics of the spray areas to protect the environment.

Moreover, environment monitoring systems with the support of WSNs take on a common architecture with multi-layers. Zhang et al. (2011) provided a general agricultural environment monitoring platform (GAEMP) based on WSN. The system architecture was defined in four layers, which were sensor node layer, gateway layer, central platform layer, and application layer. Li et al. (2009) developed a hybrid soil sensor network (HSSN) for in-situ, real-time soil property monitoring. The architecture included three layers as an in-field local wireless sensor network, a long range cellular communication network, and a web portal layer. In general, the architecture of an environmental monitoring system with the support of WSN technology can be divided into three layers, which are, from the bottom to the top, a local wireless sensing and actuating layer, a communication layer, and a data analysis and intelligence demonstration layer. The local wireless sensing and actuating layer, comprised of various wireless sensor and actuator nodes, measures factors in the environment, transmits data to the communication layer, receives commands from the communication layer, and finally change the environment by actuators. In between, the communication layer, with multiple data encapsulating, buffering, routing, and transmitting devices, forms a virtual link to switch data between the bottom layer and the top layer. And the data analysis and intelligence demonstration layer consists of data storage media, application servers, and intelligence exhibition platforms. It saves the data, converts the data into intelligence, presents the intelligence to end users, and makes decisions automatically or based

on users' input to change the environment. The principle of multi-layer design reduces the coupling effect between different layers.

Although WSNs for environmental monitoring are becoming more common, in the absence of affordable and reliable in-situ commercial sensors, measurements still rely on proxy measurements (Huma et al. 2013). Therefore, there is still a desire for more accurate, inexpensive, and miniature sensors with a long field life.

2.2 Radio propagation in dense vegetative area

The radio propagation model is a mathematical formulation, which quantifies the characteristics of radio waves as a function of carrier frequency, transmitter-receiver (T-R) separation distance and other related factors (Rappaport and Milstein 2002). These models typically predict path loss within a link or around an effective coverage area of a transmitter. Propagation models can be categorized into two main types – the empirical (statistical) models and deterministic models. Empirical models, such as the COST-231 Hata model (COST Action 231 1991), are developed based on extensive measurements and statistical properties, and often used in practice because of low cost and simplicity with acceptable accuracy. On the other hand, deterministic models, such as the ray-tracing propagation model (Athanasiadou, Nix, and McGeehan 2000), make use of physical laws, which require numerous data of site profile and calculation to derive more accurate models.

Propagation models to predict the signal strength of mobile communication systems in the outdoor environment have been studied extensively. Examples included the Longley-Rice model (Rice et al. 1967), Edwards-Durkin model (Edwards and Durkin 1969), Okumura model (Okumura, Hmori, and Fukuda 1968), Hata Model (Hata 1980), Walfisch and Bertoni Model (Walfisch and Bertoni 1988), and COST-231 Hata model. Most of these models are classic models used to conduct practical link budget design. During the past few years, researchers have developed new path loss models to optimize these models based on measurements and regression analyses (Mardeni and Siva Priya 2011). However, applications of these models have constraints on carrier frequency, antenna heights, and terrain profiles (Mardeni and Yih Pey 2012).

In the case of wireless fixed access systems, vegetation, such as foliage of trees and bushes, may exist individually or in patches. These obstructions may give rise to both absorption and scatter of radio waves. An analytical study performed by Caldeirinha (2001) showed that the

trees present in the point-to-point radio path could influence the level of the received signal. The signal was attenuated by free space propagation, scattering, which resulted in lateral contributions to the received signal, and depolarization of the incident waves. Baker (2010) investigated the impacts of vegetation on cellular signal strength based on observations collected over a nine month period. He found that the signal strength was stronger in the winter than in the fall, and a significant difference existed between the deciduous and the evergreen vegetation.

Models for predicting the attenuation of radio waves by trees reported in the literature up to 1982 were summarized in Weissberger's report (Weissberger 1982). Information contained in the report falls into two broad categories: the available prediction models and the available measurements data. In the past 30 years, many accurate models were developed to characterize the effects of vegetation on propagation and to predict the signal attenuation due to the vegetation. Empirical models include the Modified Exponential Decay Model (Weissberger 1982), the ITU-R Model (Stephens, Al-Nuaimi, and Caldeirinha 1998), and the COST 235 Model (Hall 1993). The main advantage of these models lies in the simplicity of the mathematical expressions, while the drawbacks are their dependence on specific measured data and their failure to involve physical principles of signal attenuation.

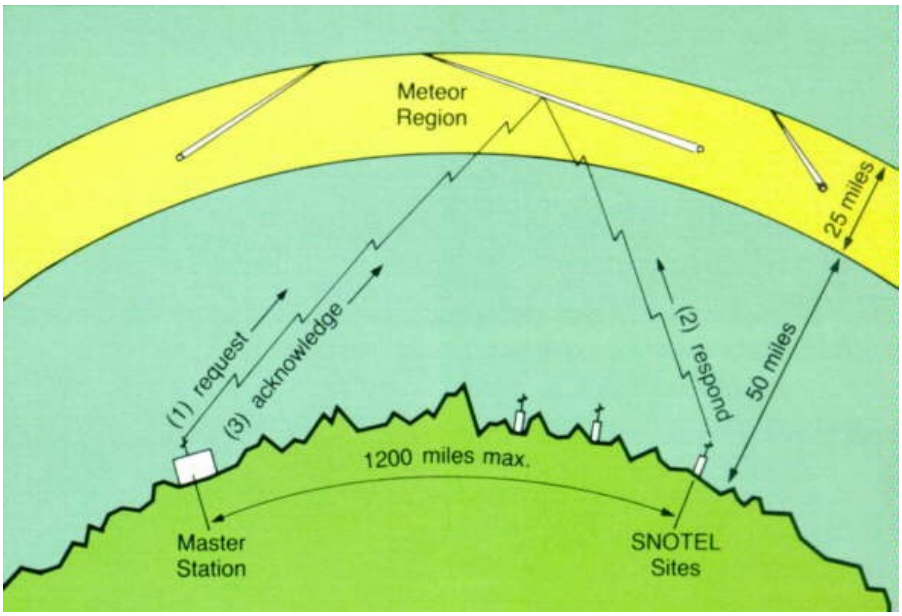
Semi-empirical models are based on knowledge of the qualitative behavior of absorption and scatter in homogenous scattering media, without considering the physical principles of signal attenuation. This category of models was introduced by formulas that give best fits to measured data (Seville 1997; Paulsen and Seville 2000) . In contrast to the above mentioned models, analytical or theoretical models offer an insight into the physical processes involved in radio wave propagation through vegetation, based on theories like the Geometrical and Uniform Theory of Diffraction (GTD/UTD) (Matschek and Linot 1999), the Radiative Energy Transfer Theory (RET) (Al-Nuaimi and Hammoudeh 1994), and the propagation channel characteristics (Koh 2002; Wang 2006). Based on the existing models, the United Kingdom Radio-communication Agency set up a project (Rogers et al. 2002) aiming to give more separable results for the differing mechanisms of propagation through the vegetation by increasing the number of measured species, geometries, and frequencies. Their ultimate goal was to develop a more complete and predictable model that includes vegetation parameters characterizing the density of vegetation.

Propagation models research in the WSN area also emerged during the past few years. Li et al. (2010) evaluated the relationship between path loss and packet delivery rate of a 2.4 GHz in-field WSN system and provided models to predict path loss in wheat field. Li et al. (2009) established a path loss model to predict 2.4GHz wireless signal transmission in residential houses to guide configuration of reliable wireless network application in heating, ventilating and air conditioning (HVAC) system control. Darr and Zhao (2008) developed a two-dimensional path loss prediction model in poultry layer facilities, which was able to predict 86% of the system variability and was able to predict an average error of -0.7 dB from the measured path loss values. No propagation models for radio propagation in a dense vegetative area have been found.

2.3 Meteor Burst Communication

Meteor Burst Communication (MBC) is an alternative transmission method to cellular service for communication in long distance. As billions of meteors have been caught by the earth's atmosphere every day, they burn up and create meteor trails. Although the trails last only a few seconds, the time is long enough to reflect radio waves (Healy, Shaw, and Litko 1989). The MBC technology was first developed by amateur radio operators in the 1940s and was explored by the U.S. military in the 1950s. By utilizing the ionized trails of gases left from the entry and disintegration of the meteors, people are able to create communication networks between different points on Earth by reflecting signals off the disintegrating meteor trails of ionized gases and back to a receiver station, located up to 1600 kilometers away (Brown 1985). Figure 2.1 shows the working principle of MBC.

Figure 2.1 Meteor Burst Communication (2006)



Advantages of the MBC system include low equipment complexity, low running cost, low probability of intercept and anti-jam (Oetting 1980). The signal transmission is much less affected than the cellular service by various natural and man-made ionospheric disturbances, including nuclear explosion. The system is particularly suited to long range, low rate data acquisition applications, although the ionized trails exist for only a few seconds (Leader 1974).

A typical MBC system is generally organized as follows: a large master base station with approximately 5,000 watts of radio power and a large antenna array that is used in conjunction with any number of remote sites, each with a 100-watt radio and a directional antenna (Cumberland, Valacich, and Jessup 2004). The master base station poll all remote sites at first. Then the remote stations respond by sending collected data and messages to the master station. The different meteor trails effectively act as a multiplexer, sequencing the responses. Each response is checked for completeness by the master station and an acknowledgment is transmitted.

Over the past 40 years, the U. S. government and several companies have developed many MBC systems in message (email or paging) backup, data acquisition and remote sensing, broadcasting, and vehicle tracking. The Natural Resources Conservation Service (NRCS) of the U.S. Department of Agriculture (USDA) set up the SNOTEL (SNOwpack TELemetry) system using the MBC technology to collect snowpack and related climatic data since 1970's (Johnson 1987). The system relays the data from almost 700 automated stations in remote locales,

including 25 in Alaska, to a central computer at NRCS's National Water and Climate Center (NWCC) in Portland. Another MBC system named TransTrak is a two-way messaging system for the transport sector, providing communication between mobile stations in trucks and a central office for vehicle dispatch and tracking in support of fleet management systems (Mickelson 1989). The Advanced Research Projects Agency (ARPA) established an Advanced Meteor Burst Test Bed (AMBTB) that operated between Charleston, South Carolina, and Verona, New York (750 miles in between), to demonstrate the capabilities of multi-media communication via meteor scatter to prospective Government and commercial users (Desourdis et al. 1994).

Although the MBC technology has had a long history, its development in both theory and practice has not become a huge success. The weak points of the technology include the long delay time, low throughput, high transmission power and large antenna size. However, these drawbacks may be alleviated by appropriate network topology, transmission protocol, and routing algorithms.

2.4 Over-the-Air Programming

For WSNs deployed in remote and unattended locations, various environmental conditions often make it impossible to reprogram the sensor nodes to adapt to new network distribution and functional requirements. Thus, Over-the-Air Programming (OAP) is needed for remote maintenance and reprogramming of sensor nodes.

OAP refers to various methods of distributing new software updates and configuration settings to radio embedded devices. It enables the update without physical access to the equipment, and updates numerous devices simultaneously by broadcasting. This technology was originally developed to enable device manufacturers and network operators to deliver updated firmware for mobile handset, as in the case of new cellphone activation. The technology is also used to update firmware running in implantable medical devices, such as pacemaker and implantable medical devices (IMD) (Gupta 2012).

The OAP function has already been embedded in motes provided by different manufactures, such as the IRIS Mote (MEMSIC Inc., Andover, MA) and the PSoC (Programmable System on Chip) devices (Adly et al. 2010). The latter can also be used as a co-processor for the Motes. Several protocols have been developed for OAP in WSNs, including Deluge (Hui and Culler 2004), MOAP (Stathopoulos, Heidemann, and Estrin 2003), and MNP

(Kulkarni and Wang 2005). However, all these protocols are platform dependent. When a protocol needs to work on a different hardware platform, software modifications are required. Recent research on OAP has been focusing on protocol improvement, including reducing the need for packet rebroadcasting (Hagedorn, Starobinski, and Trachtenberg 2008), avoiding software version inconsistency (Lee and Shen 2007), and securing OAP against unauthorized and malicious reprogramming attempts (Aschenbruck et al. 2012).

Chapter 3 - IMPROVEMENT OF WIRELESS TRANSMISSION METHOD

3.1 Introduction

The three-tier WSN was deployed at three experiment sites in the country since 2009, which were Fort Riley in Kansas, Fort Benning in Georgia, and Aberdeen Proving Ground in Maryland. Although the three-tier WSNs has been proven capable of real-time monitoring of the SSC and flow velocity through a long-term field experiment (Han 2011), the performance of the WSN was restricted by several problems in the transmission method. Improvements made on the method are discussed in this Chapter.

3.1.1 Inadequate real-time performance

As mentioned before, the real-time performance in this three-tier WSN was mainly evaluated by the time delay between sampling the raw data and receiving the signal in the server. In the LWSN tier, the Mote at the sensor node sampled two raw SSC data sets every minute. Each raw data set included voltage signals from multiple LEDs in the sensor. According to the original transmission method, the Stargate, a single board computer in the MRWN tier, could buffer 12 new raw SSC data sets. Therefore, a maximum delay of six minutes could happen if all 12 raw data sets were stored in the buffer before they were finally transmitted through the MRWN. As a real-time monitoring system, this long time delay should be minimized.

When the raw data set was received by the Stargate, it would be appended with a time stamp when it was received at the Stargate and a cyclic redundancy check (CRC) code for error detection, hence becoming a data record. After the processing, the original Stargate program encapsulated three data records into a data packet, which would be relayed through the MRWN tier to the central station. The purpose of setting such a packet length was to decrease the communication frequency to save the wireless transmission power and the occupancy of the wireless channel. However, the long packet could cost longer time to be transmitted than a shorter packet, hence causing longer time delay.

Lastly, in the original program, the Stargate only sent one data packet out within a fixed 30-second cycle, even if the previous data packet has been successfully sent to the central station. This periodic control method also decreased the real-time performance.

3.1.2 Potential data loss

In the original Stargate program, the incoming raw data set was identified by the time stamp. Only those raw data sets with new time stamps would be further processed. Consequently, if unexpected condition, such as system rebooting, occurred to the Stargate, the newly received raw data sets with old time stamp would be considered as processed data, hence causing data loss.

As described in Section 3.1.1, the original program packed three data records into a data packet. When the data packet is longer, more transmission error and data loss might occur.

Additionally, in the MRWN tier, data packets were transmitted from the Stargate to the datalogger at the gateway station through serial communication. However, no acknowledgement from the datalogger was received at the Stargate. If the data packet was not transmitted to the central station successfully, the Stargate could not detect it and would send a new data packet to the datalogger. Thus, the unsuccessfully transmitted packet was overwritten in the datalogger at the gateway station, hence causing additional data loss.

Furthermore, at a gateway station, the Stargate output a data packet to the datalogger every 30 seconds. If there exist multiple gateway stations in a MRWN, the period of 30 seconds might not be sufficient for the central station to finish collecting data packets from one gateway station before starting to collect from the next one. The condition always occurred when the transmission from one gateway station to the central station cost more than 30 seconds due to the low wireless signal strength in that channel, hence influencing the other transmission link between the other gateway station at the same MRWN to the central station. If the deadline was missed before the data packet must be transmitted out from a datalogger at a gateway station, a datalogger would be overwritten by a new data packet from the Stargate, causing more data loss.

3.1.3 Limited transmission throughput

The transmission throughput was measured by the successful data records delivered per hour over a link from the gateway station to the central station. According to the original scheduling method, the Stargate in the MRWN tier send three data records to the datalogger every 30 seconds. As a result, the ideal transmission throughput over a link from the gateway station to the central station was 360 data records per hour. The Stargate in the MRWN tier received two raw SSC data sets from the sensor node every minute, and sent two SSC data

records to the datalogger after processing. Therefore, 120 SSC data records would be sent out from the gateway station every hour. By one velocity measurement every hour, 80 velocity raw data sets would be generated and sent to the Stargate to be processed into velocity data records. Therefore, a total quantity of 200 data records could be transmitted within an hour, which was lower than the maximum transmission throughput of 360 data records per hour by the original scheduling method. However, once the velocity measurement increased to four times per hour, which meant 240 extra raw velocity data records were generated, the WSN could not transmit 440 data records within an hour without changing the scheduling method.

3.2 Methodology

Based on the problems we have found on the current wireless transmission method, several changes were made to improve the performance of the WSN.

3.2.1 Shortening the Stargate buffer size and data packet length

In the new Stargate program, the buffer was reduced to save only one raw data set. Once the raw data set was received, the Stargate would process it into a data record and cleared the buffer for the incoming raw data set. With a short buffer size, the waiting time before the data records got processed was shortened.

As mentioned in Section 3.1.1, three data records were encapsulated into one data packet by the data processing program. In order to transmit as many data as possible through each transmission and to shorten the data packet encapsulation waiting time and transmission time, the new program filled a data packet with two data records. Structure of the new data packet is shown in Figure 3.1.

Figure 3.1 New data packet with two records

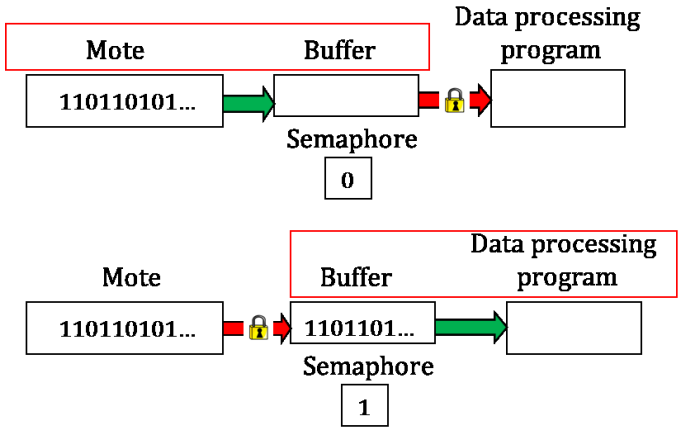
1	2	3	4	5	6	7	8	9	10
PakBusAdd	SeqNum	Day	Time	GroupID	MoteID	MoteVoltage	IR45 On	IR45 Off	ORA45 On
11	12	13	14	15	16	17	18	19	20
ORA45 Off	ORA180 On	ORA180 Off	(Reserved)	Rain Gauge	Thermocouple	(Reserved)	(Reserved)	(Reserved)	CRC(LWSN)
21	22	23	24	25	26	27	28	29	30
Day	Time	GroupID	MoteID	MoteVotage	IR45 On	IR45 Off	ORA45 On	ORA45 Off	ORA180 On
31	32	33	34	35	36	37	38	39	40
ORA180 Off	(Reserved)	Rain Gauge	Thermocouple	(Reserved)	(Reserved)	(Reserved)	CRC(LWSN)	Gateway Volt	Repeater Volt

3.2.2 Applying synchronization mechanism

Instead of using the time stamp to identify new incoming raw data, a new synchronization mechanism using semaphore was adopted. A semaphore variable was created

and was stored in a file in the CF card. The semaphore variable was initially set to "0", which indicated that no new raw data had been received and the buffer was open to the receiver Mote attached to Stargate. At this time, access of the raw data processing program to the buffer was blocked by the semaphore and the Mote could fill the buffer with new data. When the buffer was filled, the semaphore variable would be updated to "1" immediately. When the processing program observed the semaphore's toggle, it would read the data from the buffer and start to process the data. At this time, the Mote was blocked by the semaphore until the semaphore was updated to "0" again. The mechanism is shown in Figure 3.2.

Figure 3.2 Semaphore mechanism



With this mechanism, even when unexpected conditions, such as Stargate reboot and gateway station power loss, occurred to Stargate, by reading the last semaphore value from the file stored in the CF card, the data processing program would know whether the data in the buffer were processed. Thus, data loss and retransmission would be avoided.

3.2.3 Adopting feedback mechanism

Following the new transmission method, an acknowledgement signal would be sent from the datalogger to the Stargate at the gateway station, if the data packet was sent to the central station successfully. Once the acknowledgement signal was received, the Stargate would start to send a new data packet to the datalogger without waiting for time elapse until the end of 30 seconds. Otherwise, if no acknowledgement signal was received within 30 seconds, the Stargate would resend the old data packet to the datalogger at the beginning of the next 30 seconds. This algorithm helped improve the transmission throughput while reducing data loss.

3.2.4 Improvement in scheduling scheme

Originally, the Stargate in the MRWN tier sent a data packet to the datalogger every 30 seconds. After receiving the data packet, the datalogger at the gateway station would wait for a beacon signal from the datalogger at the central station to start transmitting the data packet to the LRCN tier. If the datalogger at the central station received data from the datalogger, to which it sent the beacon signal, it would keep connected with that datalogger until the data packet was received successfully. Otherwise, the datalogger at the central station would send a beacon signal to the datalogger at another gateway station for data reception. This scheduling scheme could be considered as an event driven scheme, as the transmission would start when the event (the data packet was ready at the gateway station before the beacon signal was received from the central station), occurred. As mentioned before, there were two weak points in the original scheduling scheme. Firstly, as multiple gateway stations may be deployed within this tier, the data packet received in the datalogger may not be sent to the central station within 30 seconds before overwritten by a new data packet sent from the Stargate, hence causing data loss. Secondly, the transmission throughput was constraint by the fixed time period adopted in the Stargate program. To improve the scheduling, a time driven scheduling scheme was developed. In addition, the original event driven scheduling scheme was improved.

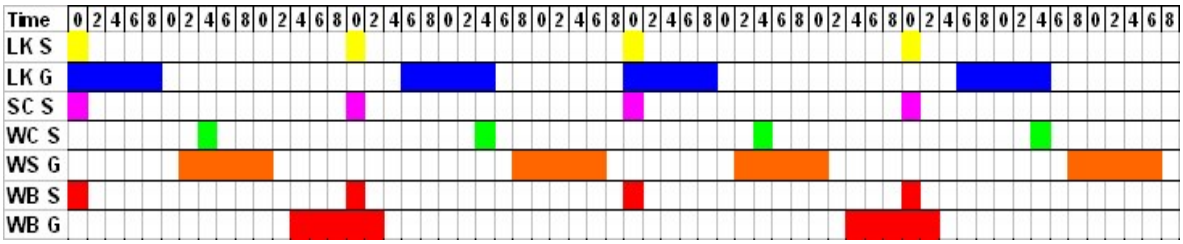
3.2.4.1 Time driven scheduling

In this section, the WSN established at the Ft. Riley experimental site was used as an example. As shown in Figure 3.3, four sensors were deployed at this site. They were located at Little Kitten Creek, Wildcat Bridge, Wildcat Creek, and Silver Creek. These sensor nodes are denoted as LK-S, WB-S, WC-S, and SC-S, respectively, in this dissertation hereafter. Among these sensor nodes, WC-S and SC-S shared a gateway station, denoted as WS-G, while the other two had their own gateway stations, denoted as LK-G and WB-G, respectively.

Figure 3.3 Map of the Ft. Riley experimental site

The time-driven scheduling method was based on a fixed task execution sequence and an equal time slot for every gateway station to transmit the data packets to the central station. A Gantt chart with a hyper-period of two minutes was shown in Figure 3.4 to demonstrate the scheduling sequence. In the chart, the colored grids in the sensors' rows represent the generation of a raw SSC data set. For each sensor, one raw SSC data set was generated every 30 seconds. The Stargate processed each received raw data set into a data record, and send a data packet comprised of two data records to the datalogger at the gateway station. In the rows for the gateway stations, the beacon signal from the central station would be sent to each gateway station at the beginning of a fixed time slot of 10 seconds. If the data packet was ready to be sent at the gateway station, the central station would start receiving the data packet in the 10-second time slot. Otherwise, the central station would stay in an idle state for 10 seconds and send another beacon signal to the next gateway station. As shown in the Gantt chart, five time slots were provided to three gateway stations. Among the time slots, two were assigned to WS-G, as there were two sensor nodes under that gateway station. Two were assigned to LK-G, because an extra 80 velocity raw data sets would be generated every hour at the Little Kitten Creek sensor node. The clock on the dataloggers at the gateway station didn't need to be synchronized. It is because, even if the data packet was not ready at the gateway station before it received the beacon signal, the packet could be sent in the next time slot.

Figure 3.4 Gantt chart of time-driven scheduling



Controlled by this purely time-driven scheduling method, the deadline would not be missed. However, the data transmission throughput of the WSN system was still limited by scheduling. For example, if the frequency for velocity measurement increased to four times per hour, 240 extra data records would be sent. In that case, the fixed time-driven scheduling would not allow all the data records to be transmitted. Moreover, because this scheduling scheme was based on fixed sensor node and gateway station distributions, it would not allow changes in these distributions.

3.2.4.2 Event and time driven scheduling

This scheduling method mixed the event-driven with the time-driven scheme. As long as an event (a data packet was ready in the datalogger at the gateway station before it received the beacon signal from the datalogger at the central station) occurred, the datalogger at central station would keep the connection with the gateway station until the data packet was received successfully. After sending the data packet successfully to the datalogger at the central station, the transmitter datalogger at the gateway station would receive an acknowledgement signal from the central station and send another acknowledgement signal to the Stargate. With that signal, a new data packet could be sent immediately from the Stargate to the datalogger. However, if the central station could not receive the data packet from the gateway station within 30 seconds, the connection between them would be terminated, so that another gateway station could send its data packet before missing the deadline. Thus, this mainly event-driven scheduling scheme was combined with time-driven. This combined event and time driven method improved the network throughput and reduced the data loss.

3.3 Results

3.3.1 Real-time performance

After applying the new transmission method, both the raw data buffer size and the data packet length were reduced. As a result, the lag between the raw data generation and the signal reception at the server was shortened from the minute to the second level. According to the query results from the database before the new method was applied, it took more than a minute to observe the latest received signal. In addition, with the use of the acknowledgement signal and the optimized event and time driven scheduling method, the latency of each data packet was narrowed to within 15 seconds.

3.3.2 Data loss rate

The data loss rates within a one-year period before and after the program updates for all three experimental sites are summarized in Table 3.1 and Table 3.2, respectively. The calculation was based on records of all the data transmitted to the database server through the cellular service. In these tables, "n/a" indicates "not available", which was caused by data loss.

Issues that caused data loss included unexpected Stargate damage, Stargate replacement, reboot of Stargate and datalogger at gateway station, errors in the datalogger program, repeater station power loss, central station power loss, changes in antenna orientation, and cellular service interruption.

At each experimental sites, there were four sensor nodes. The sensor nodes at Fort Riley site were introduced in Section 3.2.4.1. Senor nodes at Fort Benning site included Upatoi Creek North (UN-S), Upatoi Creek South (US-S), Pine Knot Creek North (PN-S), and Pine Knot Creek South (PS-S). In Aberdeen Proving Ground, the sensor nodes located at Gunpowder Dock far side (GF-S) and near side (GN-S), and Anita Dock far side (AF-S) and near side (AN-S). Sensors at the Wildcat Creek (WC-S), Silver Creek (SC-S), Gunpowder Near (GN-S), and Gunpowder Far (GF-S) were uninstalled in the year of 2011. Therefore, the data loss rates from these sensor nodes were not included.

As the SSC sensors were programed to take two samples per minute, each sensor should generate 2,880 records of raw sampled data each day. When the WSNs were working normally, data that was wirelessly transmitted from a gateway station to the database server within each

day (Q_i) was counted. The discrepancy between 2,880 and Q_i was the total number of data lost within that day. The data loss rate (DLR) was defined as follows:

$$DLR = \frac{\sum_{i=1}^{N}(2880 - Q_i)}{N \times 2880} \times 100\% \quad (3.1)$$

where N was the number of days that were counted.

Table 3.1 Data loss rate of eight sensors before new mechanism was updated

	LittleKitten Cr.	Wildcat Br.	Upatoi N.	Upatoi S.	PineKnot N.	PineKnot S.	Anita N.	Anita F.
2010-02	3.98%	2.75%	n/a	n/a	n/a	n/a	n/a	n/a
2010-03	11.01%	4.76%	n/a	n/a	n/a	n/a	n/a	n/a
2010-04	7.60%	1.25%	7.04%	0.91%	9.75%	8.24%	2.24%	3.01%
2010-05	4.93%	12.47%	5.00%	0.15%	n/a	n/a	1.76%	3.80%
2010-06	1.70%	2.65%	1.97%	0.21%	n/a	n/a	2.95%	5.30%
2010-07	1.84%	2.43%	0.96%	0.25%	0.94%	1.95%	1.71%	2.23%
2010-08	6.37%	18.60%	2.20%	0.95%	1.24%	2.23%	3.60%	6.67%
2010-09	5.80%	4.65%	0.54%	0.24%	1.74%	1.76%	7.13%	8.01%
2010-10	6.59%	3.12%	1.21%	1.19%	n/a	n/a	7.17%	6.99%
2010-11	7.35%	2.28%	0.74%	0.73%	n/a	n/a	2.11%	2.36%
2010-12	7.68%	2.02%	n/a	n/a	n/a	n/a	6.33%	6.59%
2011-01	3.99%	0.36%	n/a	n/a	n/a	n/a	n/a	n/a
Average	5.74%	4.78%	2.46%	0.58%	3.42%	3.55%	3.89%	5.00%

Table 3.2 Data loss rate of eight sensors after new mechanism was updated

	LittleKitten Cr.	Wildcat Br.	Upatoi N.	Upatoi S.	PineKnot N.	PineKnot S.	Anita N.	Anita F.
2011-02	0.70%	0.63%	n/a	n/a	n/a	n/a	n/a	n/a
2011-03	0.36%	0.61%	n/a	n/a	n/a	n/a	n/a	n/a
2011-04	n/a	n/a	1.89%	0.19%	0.40%	0.14%	5.90%	1.81%
2011-05	1.27%	1.29%	1.55%	0.83%	1.42%	0.87%	n/a	n/a
2011-06	1.94%	3.86%	1.27%	1.17%	0.77%	1.31%	0.02%	0.01%
2011-07	3.82%	4.86%	1.56%	0.50%	0.73%	1.16%	0.01%	0.01%
2011-08	3.30%	15.66%	0.82%	0.79%	0.47%	1.12%	n/a	0.01%
2011-09	3.08%	5.21%	0.74%	0.73%	0.19%	0.77%	0.81%	0.78%
2011-10	2.33%	5.35%	0.68%	0.70%	0.18%	0.82%	0.53%	0.16%
2011-11	0.71%	2.01%	n/a	n/a	0.58%	0.09%	0.19%	0.14%
2011-12	0.11%	0.34%	0.53%	0.50%	0.69%	0.54%	0.02%	n/a
2012-01	0.65%	1.03%	3.43%	2.44%	2.48%	2.30%	0.19%	n/a
Average	1.66%	3.71%	1.39%	0.87%	0.79%	0.91%	0.96%	0.42%

As shown in the tables, the average data loss rates were reduced except at the Upatoi South sensor node. An abnormal data loss occurred in January, 2012, at this sensor node, which was due to insufficient space in the Stargate CF card.

3.3.3 Network throughput

As stated previously, in the original three-tier WSN, the network throughput for each sensor was 360 data records per hour due to the scheduling method used in the MRWN tier, which satisfied the requirement of 120 sediment data records and 80 velocity data records per hour per sensor node. By using the acknowledgement signal and the combined event- and time-driven scheduling method, the three-tier WSN could handle the extra 240 velocity data records.

3.4 Discussion

After the improved transmission methods were applied to the original three-tier WSN, data received in the server was updated faster than before, and also more data records could be transmitted through the WSN. However, a trade-off needs to be considered. As we know, one of the cornerstones for the WSN system design is the energy saving method to prolong its lifetime. This is especially important for the WSN deployments in the unattended area. By using the improved transmission methods, more energy might be consumed by the improved WSN than before. Firstly, although the data packet length was shortened, the frequency of data packets transmission was increased, hence causing more energy cost. Secondly, when the acknowledgement signal was used, the data retransmission rate was potentially increased. In addition, the event and time driven scheduling method required the central station datalogger to poll the gateway station datalogger more frequently, hence consuming more power as well. Thus, a more reliable and robust energy saving design may need to be developed, although the existed energy saving method was still working properly during the field experiment.

In practice, only to monitor the SSC may not need a hard real-time system, or need to transmit so much data, but the performance improvement to the three-tier WSN is still very necessary. Because the WSN infrastructure with such performance can be applied to other environment monitoring cases, such as the forest fire surveillance and the earthquake detection, which may have special requirements of real-time performance and high data throughput.

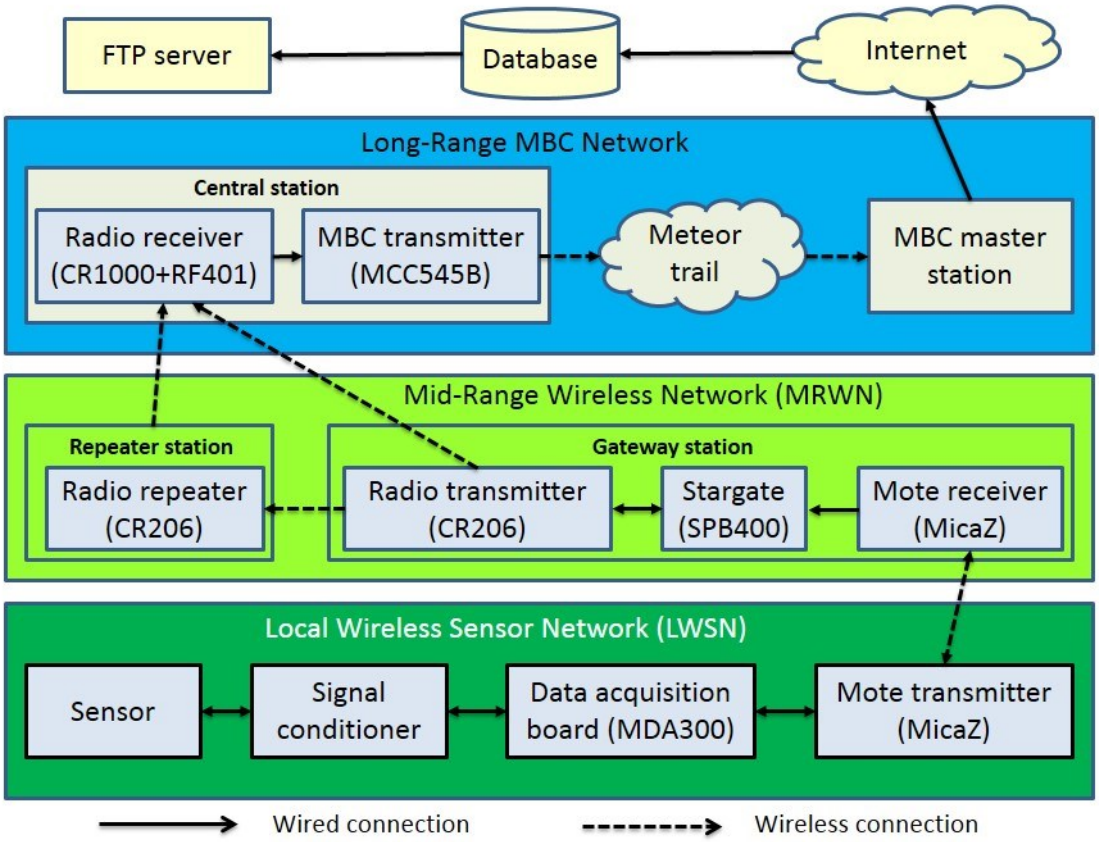
Chapter 4 - METEOR BURST COMMUNICATION IN LRCN

4.1 Introduction

In the three-tier wireless sensor network, the LRCN used commercial cellular services, including GPRS and CDMA, to transmit the data from the central station to the database server through the Internet. Although cellular services are generally reliable, issues like service stoppage, incomplete area coverage, natural disasters such as earthquake, and man-made ionospheric disturbances such as nuclear explosion, may create difficulties for continuous, real-time data collection. As an alternative for the LRCN, a meteor burst communication (MBC) system, which utilizes a medium that is not affected by these issues, was tested in this study.

A radio receiver in the long-range MBC network received data from the Gateway or repeater stations within the MRWN and further relayed it to a commercial MBC radio. The radio sent the data to a MBC master station located in Tipton, Missouri (330 km away from the central station at Fort Riley) through meteor trails. A database server at the master station would save the data from the MBC radio receiver. The other database server of Natural Resources Conservation Service (NRCS) at Portland of the U.S. Department of Agriculture (USDA) would connect periodically with the database at the master station through Internet, save the newly received data, and offer data access to MBC network service subscribers through FTP service. A block diagram of the three-tier WSN with MBC is shown in Figure 4.1.

Figure 4.1 Network architecture of three-tier WSN with MBC



4.2 Methodology

4.2.1 System components

The MBC central station included three modules: a radio receiver module, a MBC transmitter module and a power supply module.

The radio receiver module consisted of a CR1000 datalogger, a RF401 radio (Campbell Scientific Inc., Logan, UT), and a Yagi directional antenna.

The CR1000 datalogger is a precision instrument for low-power measurement applications. The CPU, analog and digital inputs, analog and digital outputs, and memory of the datalogger are controlled by an embedded operating system through a user program. The datalogger has two 9-pin connectors. One is a RS232 port for connecting to a laptop or a sensor. The other is a CS I/O port, which is designed for communications among Campbell Scientific devices. The CS I/O port can connect with PCs and communication peripherals, such as RF

devices and modems through special converters, such as SC-USB to CSI/O converter and SC105 DCE to CS I/O converter (Campbell Scientific Inc., Logan, UT). The datalogger has 4 MB memory. The program flash memory allows a maximum storage of 2 MB. The maximum operation speed is one scan per millisecond. It supports serial communication protocol, PackBus protocol, and TCP/IP protocol. The PackBus protocol is only developed for Campbell Scientific devices to share data and settings. The CR1000 uses a 12 VDC power supply with an average current drain of 27.6 mA, when measurements are taken on a single ended channel at 100 Hz sampling rate with real-time RS232 communications (Campbell Scientific 2011). Figure 4.2 shows a CR1000 datalogger.

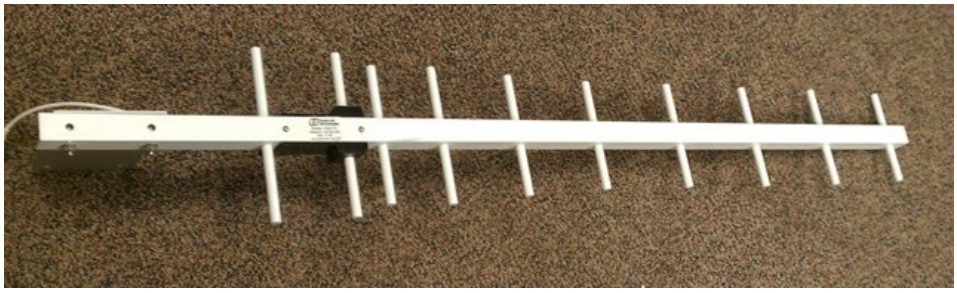
Figure 4.2 Campbell Scientific CR1000 datalogger

With two 9-pin communication interfaces, a CR1000 can enable multiple communication devices. As both interfaces are administrated by the operating system, CR1000 allows access of peripheral devices through interrupt.

In order to receive the SSC data wirelessly from the lower tier communication devices, a RF401 radio was attached to the CR1000 datalogger via the RS232 serial port. The radio uses the same wave band (915 MHz) as the CR206 at the gateway and repeater stations. The RF401 radio also uses a 12 VDC power supply with a current drain of 24 mA when receiving. Figure 4.3 shows the RF401 Radio. A 900 MHz, 14dBi Yagi directional antenna was hooked up with the radio to improve radio reception. Figure 4.4 shows the Yagi directional antenna.

Figure 4.3 Campbell Scientific RF401 Radio

Figure 4.4 900 MHz, 14 dBi Yagi directional antenna



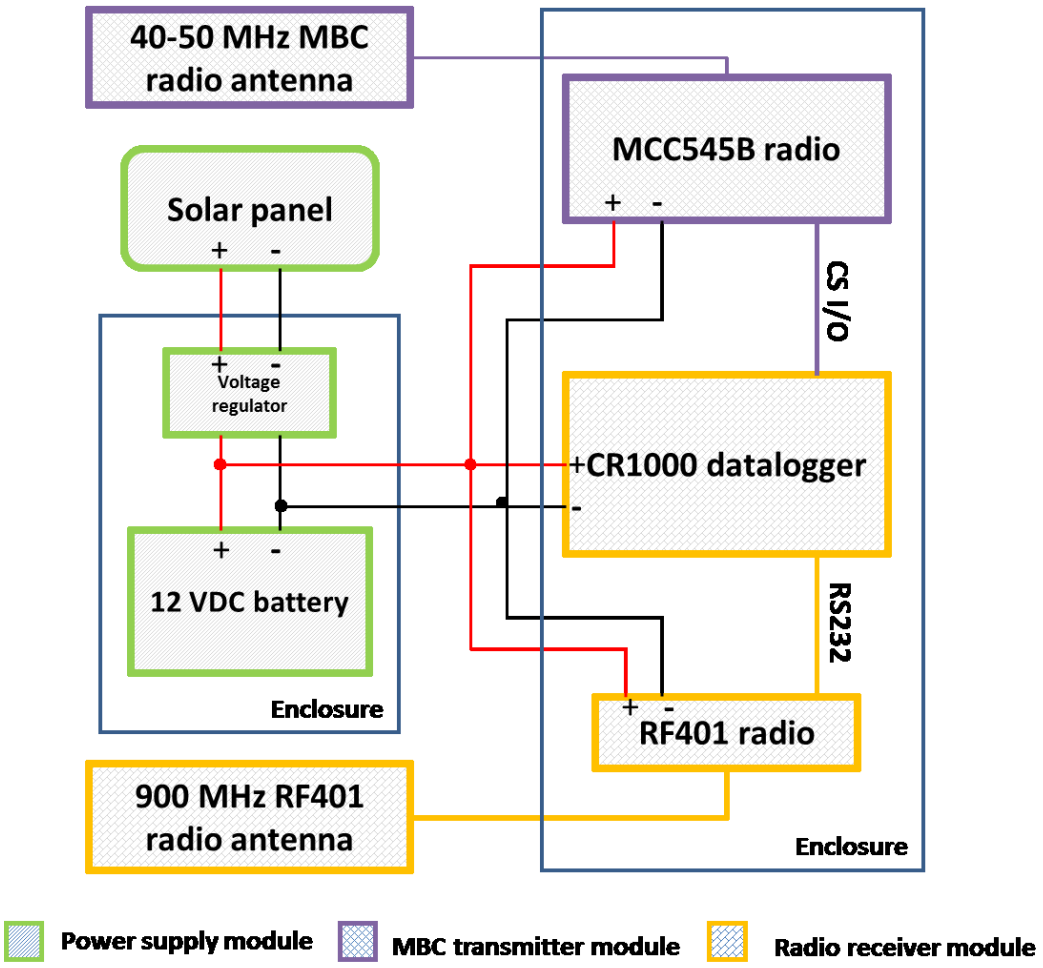
The MBC transmitter module included a MCC545B radio (MeteorComm LLC, Renton, WA) and a MBY-3 Yagi directional antenna. The MCC545B radio operates in the lower VHF band (41.61 MHz for transmitter and 40.67 MHz for receiver). It was connected with the CR1000 datalogger through the CS I/O port. The radio needed to be configured by a preloaded script file to set up the radio ID, the type of the connected datalogger, the type of information acquired from the datalogger, and the time zone where the radio is located. The script file is shown in Appendix A. The radio is powered by 12 VDC with a current drain of 80 mA at standby and 20 A for transmission for 100 milliseconds. Figure 4.5 shows the MCC545B radio. A 40 to 50 MHz, 7.65 dBi Yagi directional antenna was connected to the radio. Figure 4.6 shows the Yagi directional antenna.

Figure 4.5 MeteorComm MCC545B radio

Figure 4.6 40 – 50 MHz, 7.65 dBi Yagi directional antenna

Figure 4.7 shows the connection among all three modules.

Figure 4.7 Connections among all three modules



4.2.2 Energy harvesting

In this study, solar power was used to power the MBC central station. To calculate the total power consumption, power usage of each component needed to be figured out. Table 4.1 shows the current drains and durations of power usages of the electrical components at the MBC central station.

Table 4.1 Current drain and duration of usage of electrical components at MBC central station

Component	Current drain	Duration of power usage
CR1000 datalogger	27.6 mA (Average)	24 hours
RF401 radio	24 mA (Rx)	24 hours
MCC545B radio	20 A (Tx)	0.6 second per hour
	80 mA (Standby)	24 hours except Tx duration

Table 4.2 shows the daily power consumption of each component at MBC central station.

Table 4.2 Daily power consumption for MBC central station

Component	Daily power consumption (Wh/day)
CR1000 datalogger	7.95
RF401 radio	6.91
MCC545B radio	24.00
Total	38.86

The total power needed can be calculated using Equation 4.1.

$$P = \frac{E_{daily} \times 100\%}{S_{Avg} \times Eff_{batt}} \quad (4.1)$$

where

P = Power need (W)

E_{daily} = Daily power consumption (Wh)

S_{Avg} = Average daily insolation (hr)

Eff_{batt} = Charging efficiency of battery (%)

A typical charging efficiency for a lead-acid battery was 80%. The average daily insolation in Kansas is 4.6 hours per day. Thus, the power needed at the Little Kitten site was calculated as 10.56 W. Considering the retransmission due to unstable availability of meteor trails, the real power consumption may be higher than the calculated value. Therefore, a solar panel rated at 40 W was selected for this site.

4.2.3 Central station Software design

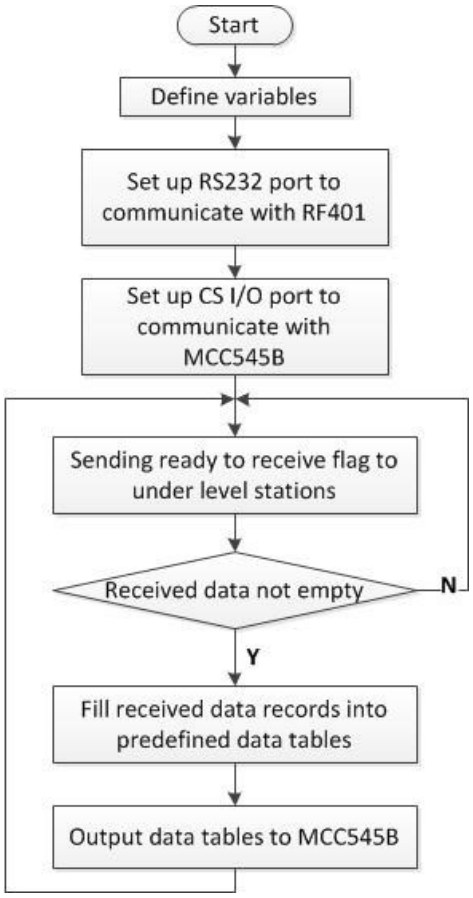
The central station software mainly included the program for the CR1000 datalogger to receive the data record from the gateway and repeater stations and to output the data record to the MCC545B radio.

The program started with the definitions of variables and data tables. When the CR1000 datalogger was initiated, it would send a flag signal of "ready to receive" through the RS232 serial port to the RF401 radio. The RF401 radio would send the signal to a station in the MRWN (a repeater station or a gateway station) following the PakBus protocol, as described in Section 4.2.1. If the data received was not empty, the CR1000 datalogger would start to fill the incoming data records into the predefined data tables. Once the tables were updated, CR1000 datalogger

would output the data records to the MCC545B radio by loading the variables in the tables through the CS I/O port.

The flowchart of the CR1000 datalogger program at the MBC central station is shown in Figure 4.8. The source code is listed in Appendix B.

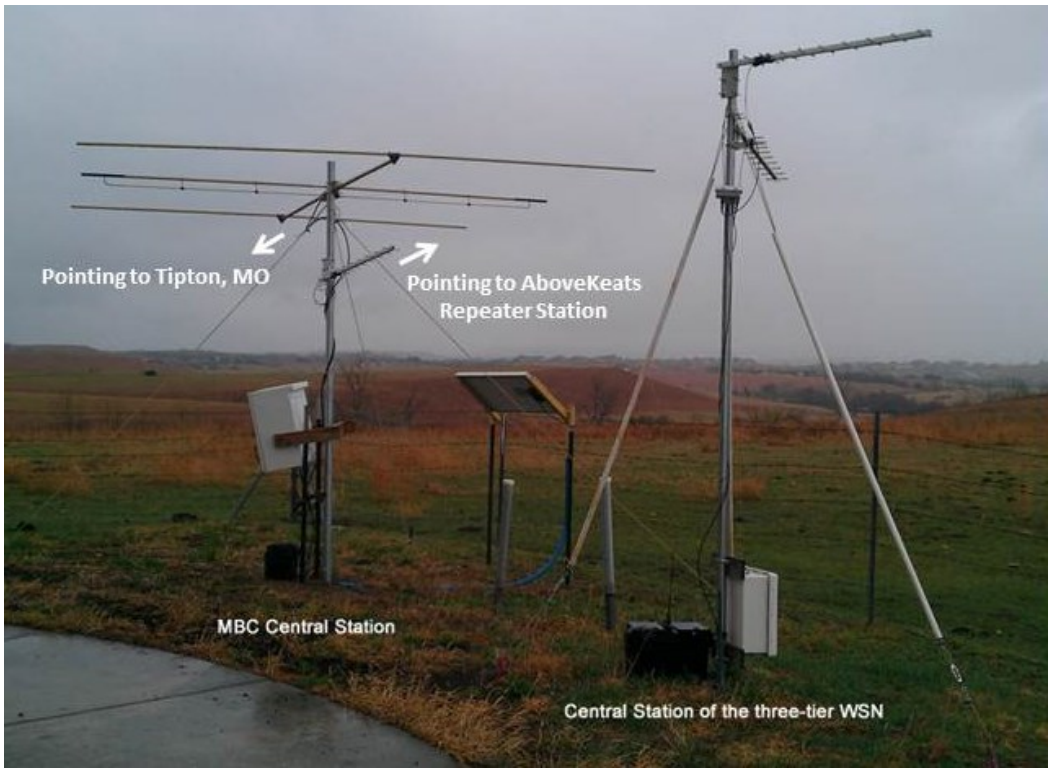
Figure 4.8 Flowchart of CR1000 datalogger program at the MBC central station



4.2.4 Deployment and installation

The MBC central station was installed three meters east to the central station of the three-tier WSN at Ft. Riley. Figure 4.9 shows the location of the two central stations. The MBC antenna was mounted at the top of a ten-foot steel conduit and was pointed to the MBC master station at Tipton, MO, which is located about 330 km to the east of Manhattan, KS. The Yagi directional antenna connected to the RF401 radio was also mounted on the conduit, heading to the direction of the repeater station at Above Keats inside Ft. Riley. The MBC central station received the SSC data from the sensor node at the Wildcat Bridge site and transmitted the data to the MBC master station.

Figure 4.9 MBC central station and the three-tier WSN central station



4.3 Results

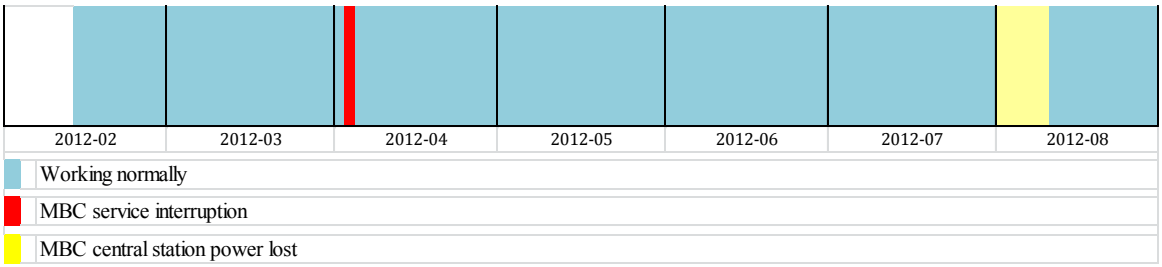
4.3.1 Working history

The MBC system was demonstrated from Feb. 15 to Aug. 31, 2012. Table 4.3 summarizes the statistics related to the working history, including the uptime and downtime for the MBC system. The issues that caused the downtime included MBC service interruption due to the service stoppage from the MBC master station at Tipton, MO (Apr. 3 – Apr. 4, 2012) and MBC central station power loss due to the weak battery (Aug. 1 – Aug.10, 2012) as shown in Figure 4.10.

Table 4.3 MBC system working history

Demonstration period (Days)	Uptime (Days)	Downtime (Days)
199	187	12

Figure 4.10 MBC system working history and issues



4.3.2 Data packet transmission duration

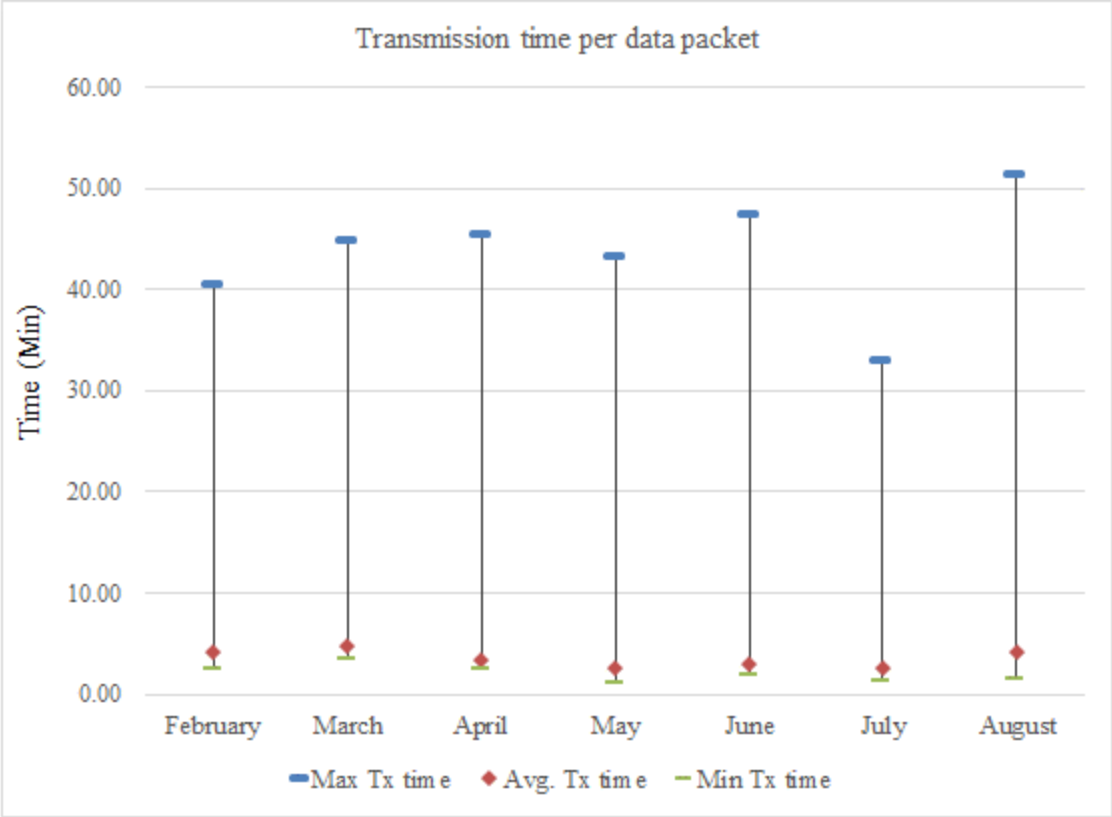
The raw data downloaded from the NRCS FTP server included four parts: header, data body, receiving time, and transmitter ID, which are shown in Figure 4.11. Before the transmission, each complete SSC data packet, as shown in the data body part, was divided into four segments, with 10 four-byte variables in each segment. The MBC radio at the central station would send the four segments to the master station in turn. When a data segment was received at the master station, the server program would add a header at the beginning of the data segment, using "G01" to "G04" to mark the receiving sequence. A time stamp and a transmitter ID would be added to the end of the data segment, indicating the receiving time and the transmitter radio ID respectively.

Figure 4.11 MBC raw data format

Header	Data body	Rx time	Tx ID
G01,S11,10/15/2012	+3.000,+0.000,+289.0,+5772.,+0.000,+3.000,+3304.,+236.0,+0.000,+101.0,	10/15/2012,00:12:22,	06052
G02,S10,10/15/2012	+0.000,+320.0,+0.000,+0.000,+0.000,+6.000,+289.0,+0.000,+0.000,+0.000,	10/15/2012,00:12:22,	06052
G03,S10,10/15/2012	+289.0,+0.000,+125.0,+3.000,+3304.,+236.0,+0.000,+100.0,+0.000,+293.0,	10/15/2012,00:13:40,	06052
G04,S10,10/15/2012	+0.000,+0.000,+0.000,+5.000,+0.000,+0.000,+0.000,+0.000,+12.33,+12.74,	10/15/2012,00:13:40,	06052

The duration of transmission for each data packet can be calculated by subtracting the time stamp value of the first segment from the time stamp value of the last Rx time stamp of each data packet. The minimum, maximum, and average transmission time per data packet in each month during the demonstration period is shown in Figure 4.12.

Figure 4.12 Transmission time per data packet



From the figure we know that maximum transmission time per data packet in each month was from 33 to 51 minutes. The average time to transmit one data packet by MBC was less than five minutes. Therefore, setting the data packet transmission rate to one data packet every five minutes should guarantee all the data packet to be received at the master station.

4.3.3 Data loss and transmission error rate

As the sampling rate for the SSC sensor was two samples per minute, each sensor generated 2,880 samples per day. However, considering the slow transmission rate of MBC, the data packet transmission rate was set to one data packet every five minutes. Thus, the total number of samples received in the database from the NRCS FTP server should be 288 each day. When the MBC system was working normally, the number of samples received in the database (Q_i) was counted. The discrepancy between 288 and Q_i was the total number of data lost within the day. Since a part of the data loss was attributed to the LWSN, this part of data loss should be removed from the total data loss. Thus, the data loss rate of LWSN in percent ($DLR_{i,LWSN}$) within the day was applied in the Equation 4.2.

$$DLR = \frac{\sum_{i=1}^{N}(288-Q_i)}{\sum_{i=1}^{N} 288\times(1-DLR_{i,LWSN})} \times 100\% \quad (4.2)$$

Thus, the daily average data loss rate of MBC system during the entire demonstration period can be calculated using Equation 4.3.

$$DLR_{avg} = \frac{\sum_{i=1}^{N} DLR_i}{N} \times 100\% \quad (4.3)$$

The transmission error was judged by the CRC value. When the MBC raw data were received, the server program would trim the raw data and compute a CRC value based on the SSC data section. The new CRC value would be compared with the CRC value computed in the Stargate at the gateway station. If they did not match each other, a transmission error was detected.

The data loss rate and transmission error rate during the demonstration period are shown in Table 4.4. From the literature review, we found that Fukuda et al. (2003) performed experiments on MBC in the Antarctic and reported that about 60% of the generated data packets were constantly transferred to the MBC master station within two hours delay. Comparing the performance of our MBC system with the references and considering the advances in the MBC technology in the last decade, the data loss rate and transmission error rate is relatively low.

Table 4.4 Data loss rate and transmission error rate of MBC

Item	**Month**							**Average**
	Feb.	Mar.	Apr.	May	Jun.	Jul.	Aug.	
Data loss rate	0.69%	0.69%	0.69%	1.39%	1.39%	2.08%	0.69%	1.09%
Transmission error rate	9.30%	4.89%	3.94%	2.30%	2.78%	4.17%	3.48%	4.41%

4.3.4 SSC data comparison between MBC system and cellular network

The SSC data received from the MBC system were compared with the data received from the cellular network. The SSC signals included six voltage values measured by three light emitting diode (LED) receivers, which received light from 45 degree to an infrared LED transmitter (IR45), from 45 degree to an orange LED transmitter (ORA 45), and from 180 degree to an orange LED transmitter (ORA 180) respectively. Two values were recorded when each LED transmitter was turned on and off. The SSC voltage signals were derived from the

subtraction of the on and off voltage values. Figure 4.13 to Figure 4.15 show the comparison of three voltage signals between the cellular network (CN) and the MBC system in March, 2012.

Figure 4.13 IR45 On-Off signal comparison between MBC system and cellular network in March, 2012

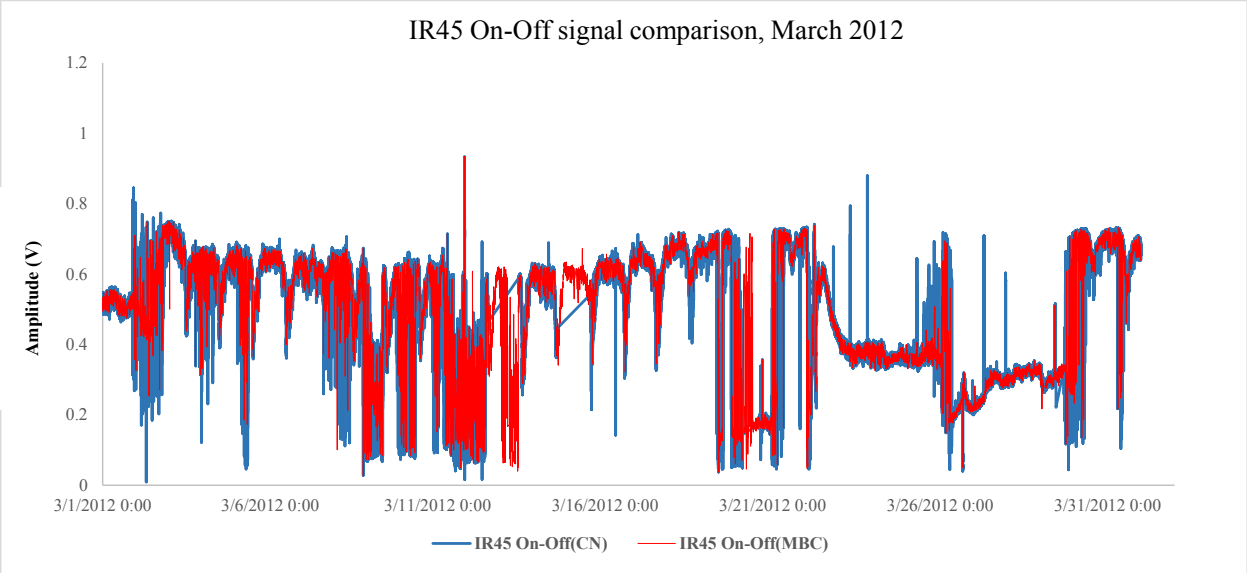


Figure 4.14 ORA45 On-Off signal comparison between MBC system and cellular network in March, 2012

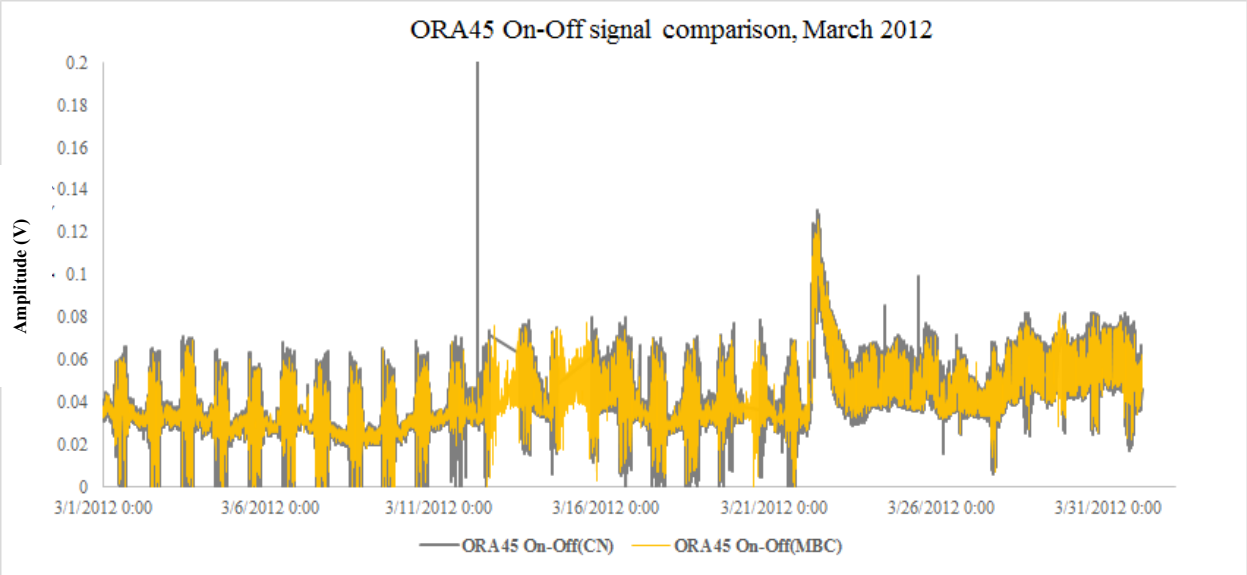
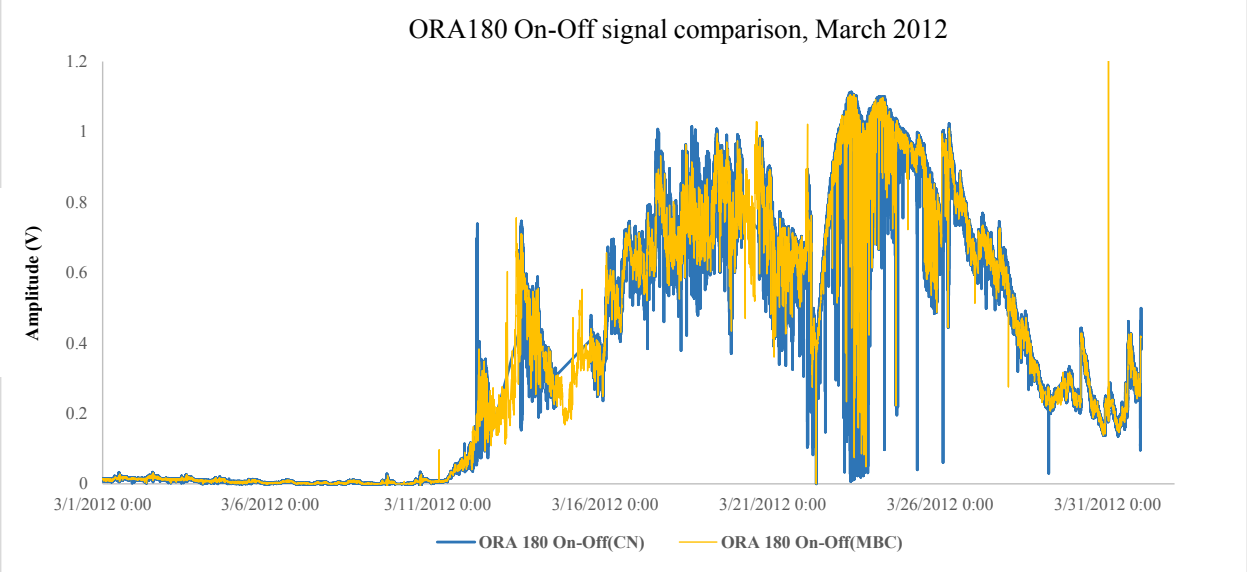


Figure 4.15 ORA180 On-Off signal comparison between MBC system and cellular network in March, 2012



From the figures, firstly, we can see that signals transmitted by the CN and the MBC system followed the same variation. This must be true because the MBC central station and the CN central station were transmitting the SSC data from the same gateway station. As the time stamp of the data was added at the gateway station but not at the central station, the signal values collected at the same time should be identical in the figure, even if the signal transmitted by the MBC central station arrived in the server later than by the CN central station. Secondly, we can roughly see that data points transmitted by the MBC system were sparser than those by the CN system, because the SSC data transmission rate of the MBC system was one record every five minutes rather than one record every minute of the CN. In addition, we can see there were some transmission gaps, such as the one around March 16, of the signals transmitted by CN in the figure, while the MBC system was still transmitting data back to the server.

Although the figures already show the same variation followed by the signals from the CN and the MBC system, it is still necessary to quantify how the signals transmitted by two systems were matched. Firstly, the numbers of data records successfully received in the MBC server from March to July of 2012 are listed in the first row of Table 4.5. Secondly, searched by the time stamp, the data record collected at the same time needs to be acquired in the CN server, as listed in the second row of Table 4.5. Among all the records found with the same stamp in the CN server, if signal values are the same to the record's values in the MBC server, it will be

counted as a matched record. By this means, the numbers of matched records found in the CN server from March to July, 2012 are listed in the third row of Table 4.5. The matching rate was calculated by dividing the number of matched records in the CN server by the number of records found with the same time stamp in the CN server.

Table 4.5 Records comparison between MBC system and cellular network

Item	Month				
	2012-03	2012-04	2012-05	2012-06	2012-07
Records received in the MBC server	8739	8004	8389	8254	8490
Records found in the CN server with the same time stamp	7754	6788	7470	6794	7472
Same records matched in the CN server	7597	6680	7426	6475	7339
Matching rate	97.98%	98.41%	99.41%	95.30%	98.22%

According to the table, possible reasons why records with the same time stamps could not be found in the CN server were: First, it was possible that the cellular network was out of service during the MBC testing time, hence causing data loss in the CN server. Second, it was possible that the CN server in the lab was off-line due to the rebooting or network issues, which also caused data loss. The matching rates were all over 95%, indicating that records with the same time stamps from two servers were basically matched. The unmatched records were probably due to data transmission errors occurred in the CN transmission or errors occurred in the MBC transmission, consequently causing different record values.

4.3.5 Material cost

Material costs of the MBC central station included costs for the radio receiver module, the MBC transmitter module and the power supply module.

Table 4.6 Material costs of the MBC central station

Component	Price
Radio receiver module (CR1000 datalogger, RF401 radio, and antenna)	$1,105.00
MBC transmitter module (MCC545B radio and antenna)	$1,560.00
Power supply module (Solar panel, voltage regulator, and battery)	$395.00
Total	$3,060.00

The costs listed in Table 4.6 do not include cost for system development, deployment, and maintenance. Although the material costs of the MBC central station at the deployment stage were higher than those of the cellular service based central station, which was about $1,180 (Han 2011), the cost for MBC system operation was lower because there was no monthly charge for the wireless service.

4.4 Discussion

The field test of the MBC system during an eight-month period demonstrated that the technology was capable of long distance communication. With proper site selection, accurate antenna positioning, and proper maintenance, the system achieved low data loss rate and low transmission error rate. However, the MBC system has a major limitation. Due to the unstable availability of the meteor trails, the MBC radio could send only one data packet every five minutes, while the cellular network was capable of sending two data packets every minute. Therefore, the lower communication throughput of the MBC system greatly limited its ability to transmit the sensor data. On the other hand, the MBC technology can be used as a long distance communication method for the other environmental monitoring systems, which only need to transmit a small amount of information, and do not need to demonstrate the data in real-time.

Chapter 5 - OVER-THE-AIR PROGRAMMING IN LRCN AND MRWN

5.1 Introduction

After the three-tier WSN was set up, the maintenance cost became a major issue. As most of the wireless devices were installed in unattended areas, maintenance of these devices, especially during or after severe weather conditions such as thunderstorms and snowstorms, was very difficult. Moreover, on-site hardware and software overshooting and updating required multiple field trips of experienced technicians, which resulted in great labor and travel costs. During the period from 2009 to 2010, the KSU research team had to travel to the Fort Benning and APG sites at least twice a year. The maintenance work included updating the software, cleaning the SSC sensors, installing or replacing new sensors, checking hardware connections, and maintaining the power supply.

One way to reduce the need for on-site visit was to configure or re-program the wireless devices remotely through over-the-air programming (OAP). With the OAP technology, software of the communication devices could be updated remotely without changing the WSN architecture and interrupting normal signals transmission.

The existing three-tier WSN basically used a one-way communication scheme, transmitting sensor signals from the sensor nodes to the server. When OAP is achieved, ideally, software update can be sent from the server down to the sensor nodes. In that case the WSN would become a bidirectional communication network, as shown in Figure 5.1.

Figure 5.1 A bidirectional three-tier WSN with OAP enabled

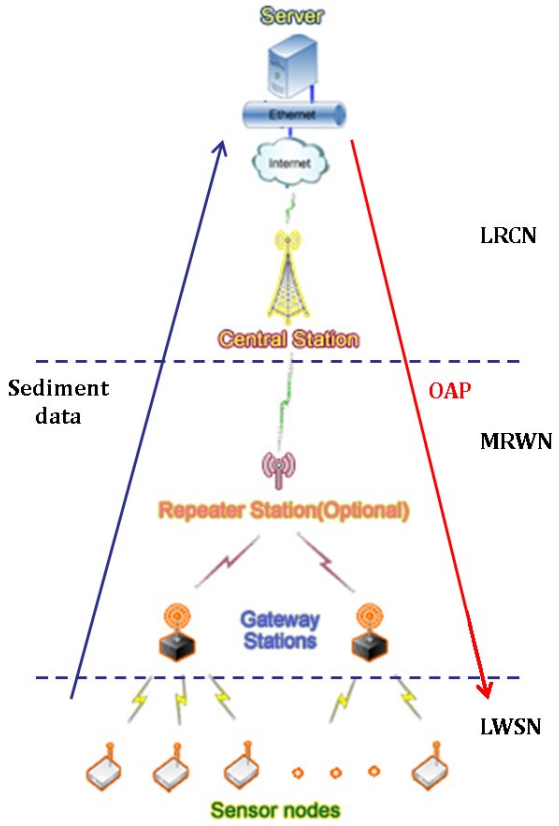
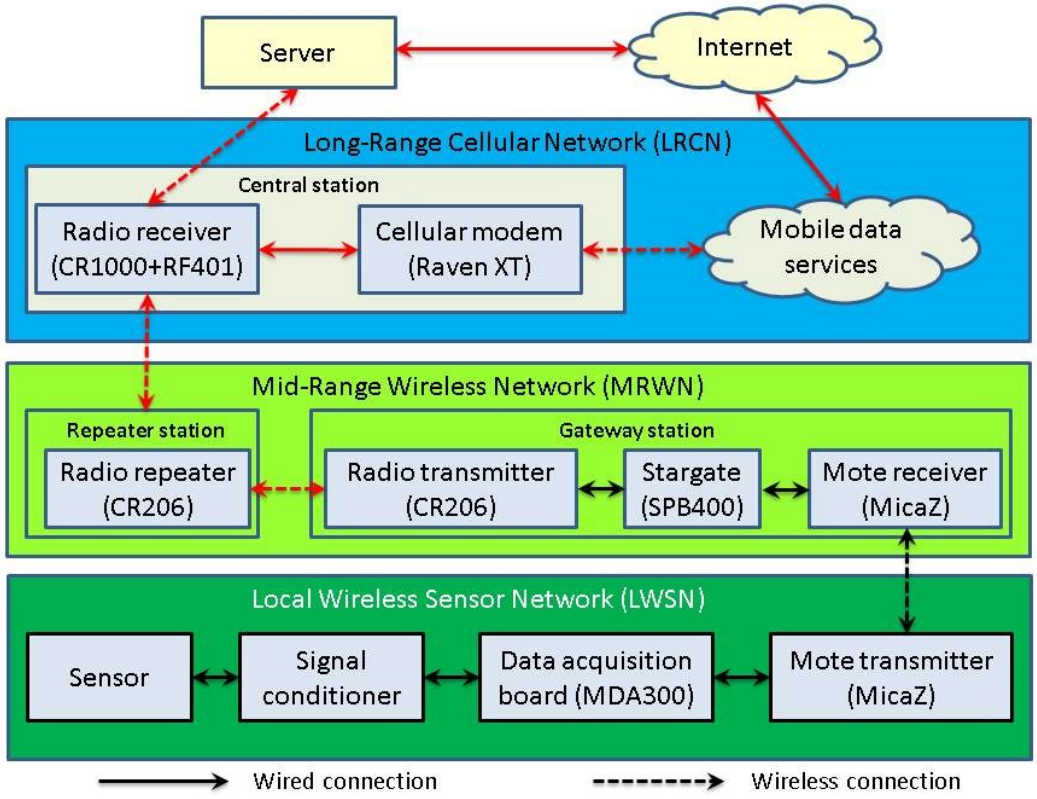


Figure 5.2 shows the architecture of the three-tier WSN with OAP implemented. A comparison between Figure 5.2 and Figure 4.2 shows that the addition of OAP did not change the architecture of the three-tier WSN. However, the added OAP feature enhanced the communications in the MRWN and LRCN tiers to bidirectional, as indicated by the red arrows in Figure 5.2.

Our experience with maintaining the WSN has shown that, the most frequently updated software has been the program for the Stargate at the gateway stations and the program for the radio receiver at the central station. As shown in Figure 5.2, the Stargate bridged the communication between the two Motes in 2.4 GHz with the communication between the radios in 915 MHz. The radio receiver at the central station bridged the 915 MHz radio communication with the cellular communication. The program for the Stargate had the functions of reading the raw data sets from the Mote receiver, trimming and formatting the raw data, packing the raw data sets into the data record, and outputting the data record to the datalogger at the gateway station for Mid-range transmission. The program for the radio receiver at the central station scheduled the data packet transmission within the MRWN tier (from the Gateway or repeater station to the

central station). It also stored the information on distributions of the Repeater and gateway stations in the MRWN. Thus, the programs for the Stargate and the radio receiver at the central station were most frequently modified. The experiment on OAP application is therefore conducted on these two devices.

Figure 5.2 Block diagram of the three-tier WSN with OAP enabled



5.2 Methodology

5.2.1 Hardware and cellular service upgrading

To achieve the OAP, a system component and the cellular service needed to be upgraded. The system component requiring upgrading was the radio receiver at the central station. In the original three-tier WSN, a CR206 datalogger (Campbell Scientific Inc., Logan, UT) was used as the radio receiver. This datalogger was replaced with a radio receiver comprised of a CR1000 datalogger and a RF401 radio transceiver.

The features of CR1000 datalogger and RF401 radio have been described in Section 4.2.1. The CR1000 datalogger was selected because its internal memory can be managed by the

embedded operating system in conjunction with the user program. The datalogger is capable of integrating the discrete data received from a remote server computer into a program file. If the program file is executable, it can be loaded to the CR1000 datalogger by the command written in the datalogger's running program. If the file is not executable, it will be compiled and linked by the operating system first and then be executed by the control of the running program. With this feature, a CR1000 datalogger can reload any new program received from the server over the air. It can also save a program for a Stargate and forward the program to the Stargate at a given gateway station.

On the other hand, the CR206 datalogger used in the existing central station doesn't have a file system. All data received by the CR206 are saved in predefined tables, which cannot be assembled as a complete file to be compiled or loaded. For a CR206 datalogger, the only way to reload the program is to download the program manually and then restart the datalogger.

In the upgraded central station, a RF401 radio was connected with the CR1000 datalogger through the CS I/O port, as there was no on-board radio inside the CR1000. The radio handled the communication with radios at the MRWN tier to allow the CR1000 to receive SSC data from the sensors and to disseminate an upgraded program to a Stargate at any given gateway station.

The other component used at the central station was a Raven cellular modem (Sierra Wireless, Richmond, BC, Canada). The modem with different firmware installed could support multiple cellular systems, such as the Code Division Multiple Access (CDMA) and the Global System for Mobile Communications (GSM) cellular systems. The Raven modem was connected to the CR1000 datalogger through a null modem RS232 cable. Figure 5.3 shows the Raven XT manufactured for use on the AT&T General Packet Radio Service (GPRS) network. The modem can be connected with a remote computer through the Internet using the TCP/IP communication protocol with a static or dynamic IP address.

Although the hardware of the cellular modem was not changed, the mobile data service we subscribed for the modem needed to be upgraded. This was because that the mobile data service we subscribed only supported one-way communication - to send the data from the cellular modem to our database server. For over-the-air programming, however, bidirectional communications were needed. For this purpose, we subscribed the AT&T I2GOLD service to connect the cellular modem with the server through the Internet. With this service, a static IP

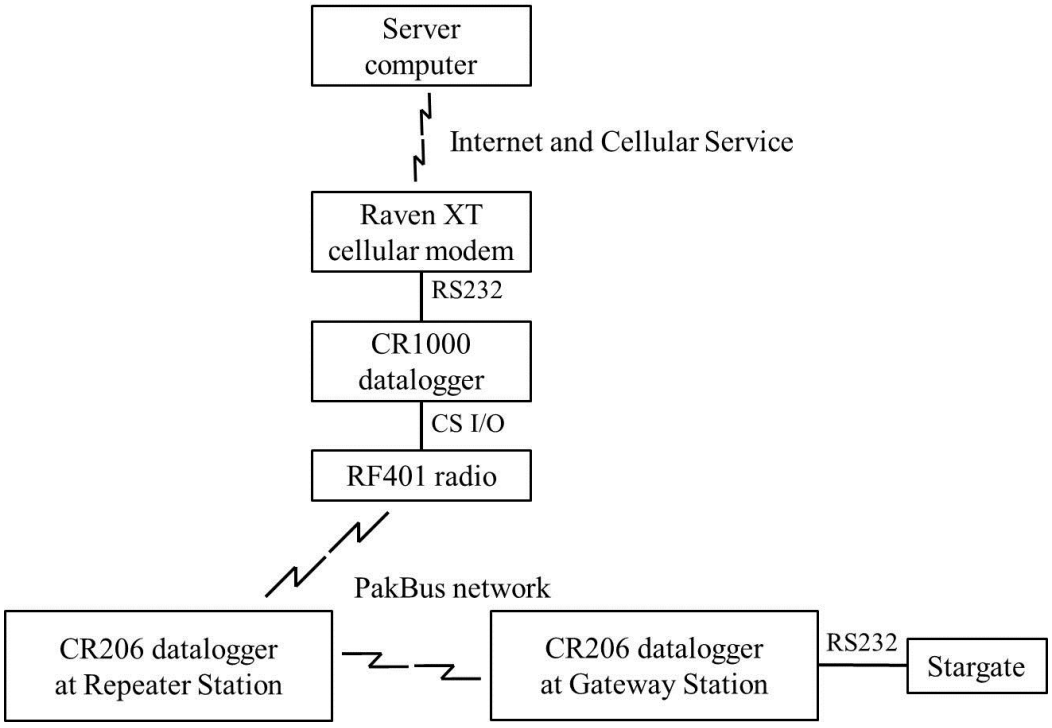
address was permanently assigned to the cellular modem and would always be valid whenever the Raven modem is connected to the Internet.

Figure 5.3 AirLink Raven XT cellular modem

5.2.2 Communication links

The communication links among the components are shown in Figure 5.4.

Figure 5.4 Block diagram of communication links for OAP



When an OAP procedure was started, the new program file would be sent from the server computer to the Raven XT cellular modem through the Internet by the mobile service. The Raven modem would then save the file in its buffer and forward it to the CR1000 datalogger

through a RS232 serial port. If the program file was for the Stargate at a gateway station, the CR1000 datalogger would send the file to the RF401 radio through a CS I/O port and the radio would then send it to the CR206 datalogger at the gateway station through the Campbell Scientific PakBus network. Finally, the CR206 datalogger would forward the program file to the Stargate through a RS232 serial port.

5.2.3 Software design

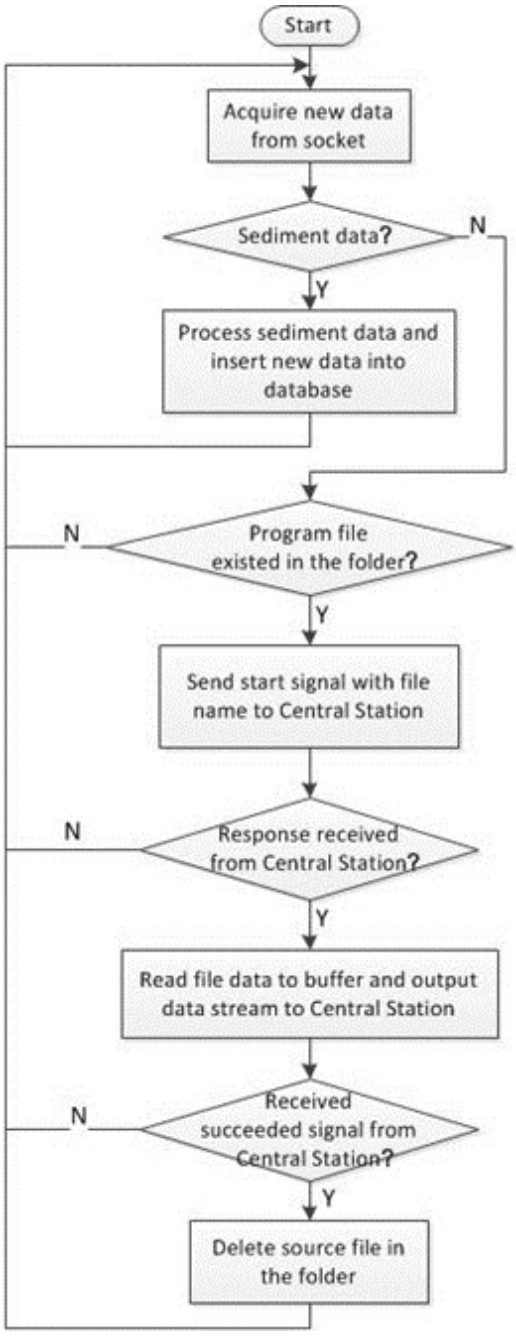
The software designed for OAP included programs for the server computer, the CR1000 datalogger at the central station, the CR206 datalogger at the gateway station, the CR206 datalogger at the repeater station, and the Stargate. All the upgraded programs kept the original functions of SSC data processing and transmitting, and then implemented the OAP functions.

5.2.3.1 Server computer program

A Java program for the server computer was developed to continuously receive data packets from the cellular modem with a designated socket. When the connection between the server computer and the cellular modem was established, the program would keep monitoring the socket. If any data packets were received at the server end, the program would process the data and insert them into the database. After finishing processing the sediment data, the program would start to check whether there existed any file in a predefined folder in the server computer to be sent to the central station. If so, a start signal plus the file name would be sent from the server to notify the central station to prepare for file transmission. When the central station received the signal, it would send a response signal back to the server indicating it was ready. If the server couldn't receive the response signal, it would keep sending the start signal to the central station until the response was received. While waiting for the start signal, the server computer would continuously process the incoming sediment data packets. Only when the response signal was received and there was no sediment data traffic, the server would begin reading the file into a buffer and send it as a data stream to the central station using the TCP protocol.

The flowchart of the Java program is shown in Figure 5.5. The source code of the programs is listed in Appendix C.

Figure 5.5 Flow chart of the server computer program

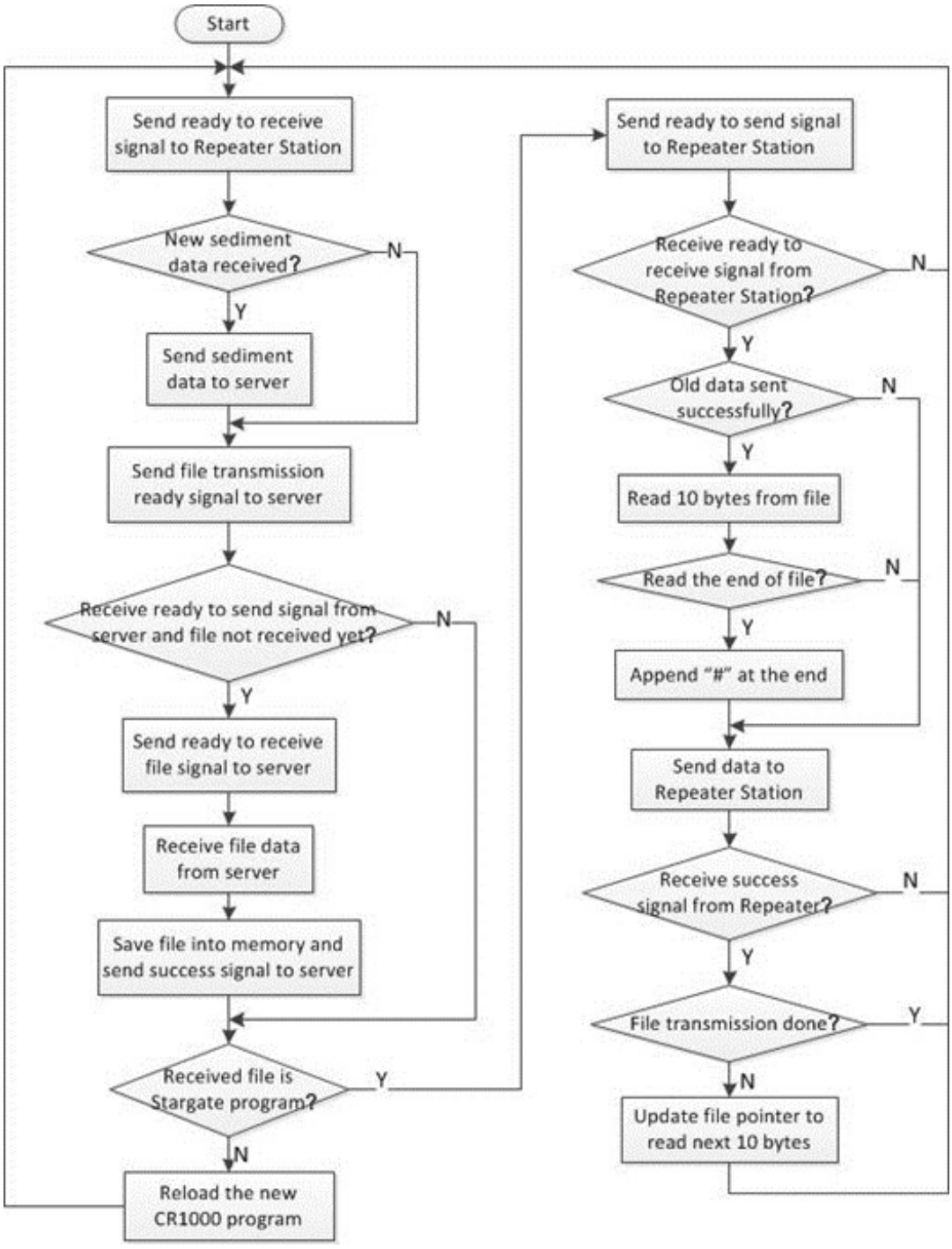


5.2.3.2 CR1000 datalogger program at central station

A CRBasic program for the CR1000 datalogger at the central station was developed to receive the sediment data from the dataloggers at the gateway stations and repeater stations, send the sediment data to the cellular modem, while receiving the program file from the server and transmitting it to the gateway station, if a program upgrade was needed for the Stargate. The

flowchart of the program is shown in Figure 5.6. The source code of the program is in Appendix D.

Figure 5.6 Flow chart of CR1000 datalogger program at central station



At the beginning of the program, the CR1000 at the central station would check if there were any sediment data packet ready to be sent from the CR206 dataloggers at either a gateway

station or a repeater station. If so, the CR1000 would receive the data packet and forward it to the cellular modem through the RS232 serial port, and the cellular modem would in turn send the data to the data server through the cellular network. When transmission of the sediment data was complete, the CR1000 at the central station would then send a signal to the server through the cellular modem, indicating it was ready for file transmission. If the CR1000 observed there were no sediment data packets ready to be sent at the beginning, it would send the signal to the data server directly through the cellular modem.

When transmission of the signal was complete, the CR1000 would check if a start signal with a file name was received from the server. If so, it would send a response signal back to the server and prepare to receive the file stream from the cellular modem's buffer. After saving the file stream with the file name in its memory, the CR1000 would send an acknowledgement signal to the server.

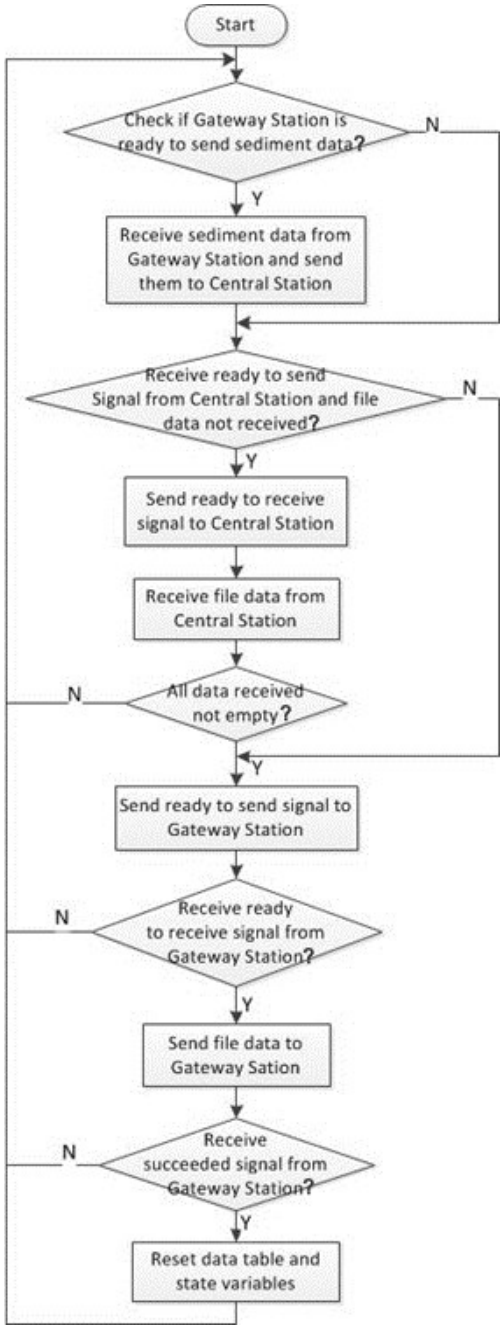
After the file was saved in the memory of CR1000, the CR1000 would check if the file was for the CR1000 program upgrade at the central station program. If it was, an operating system level command of CR1000 would be called to reload the new program to the CR1000. Otherwise, if the program upgrade was for the Stargate, the CR1000 would read segments of the program file already saved in its internal memory, 10 bytes in each segment, and transmit the segments to the CR206 at a Gateway or Repeater station in a loop. After receiving each 10-byte segment successfully, the CR206 would send an acknowledgement signal back to the CR1000. If the acknowledgement signal was not received, the CR1000 would resend the same segment in the next loop, until the acknowledgement signal was received. This retransmission mechanism guaranteed the complete transmission of the program file. When the program file was fully transmitted, a "#" symbol would be appended to the last effective segment, which was marked as the end of the file. Thus, the Stargate would know when the complete program upgrade was received.

In the loop of the CR1000 program, transmission of the sediment data packets would always be completed before the program upgrade. In addition, the CR1000 would not send the "ready to receive" signal to the server through the cellular modem, until the program upgrade for the Stargate was complete.

5.2.3.3 CR206 datalogger program at the repeater station

Functions of the original CRBasic program for the CR206 datalogger at a repeater station was to send the sediment data from the datalogger at a gateway station to the datalogger at the central station only. For OAP, the repeater station's datalogger also needed to receive the program upgrade from the CR1000 at the central station and forward it to the CR206 at a gateway station. The flowchart of the program is shown in Figure 5.7. The source code of the program is in Appendix E.

Figure 5.7 Flow chart of CR206 datalogger program at the repeater station



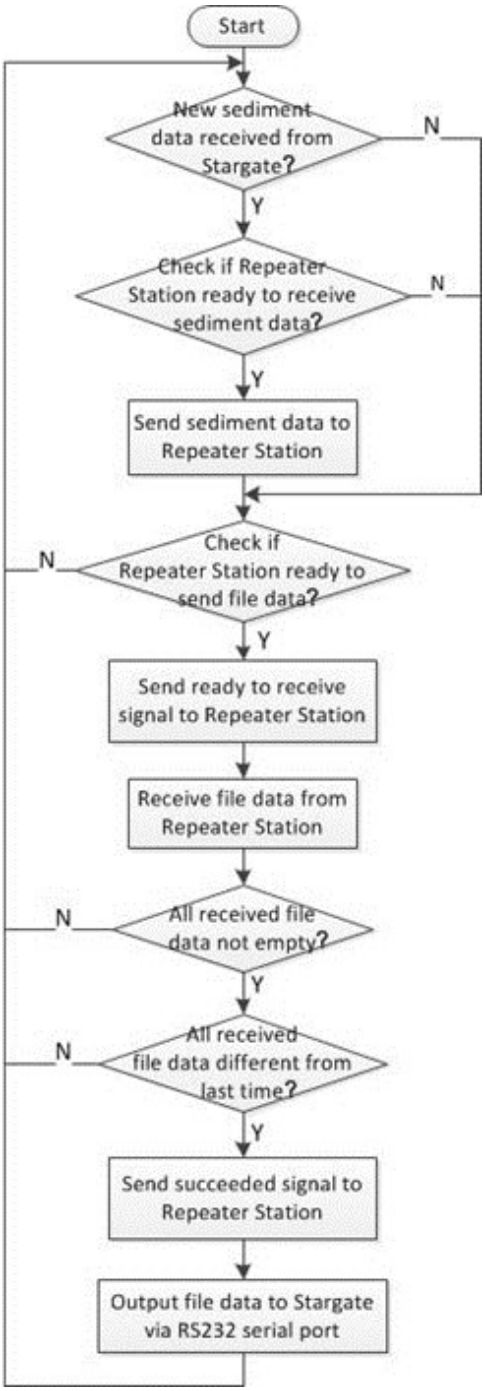
Like the program for the datalogger at the central station, the program for the repeater station would make sure that the sediment data was successfully transmitted at the beginning of each loop. During the next step, the program would check if the CR1000 at the central station was ready to send a program upgrade. During the transmission of the program upgrade, the repeater station would save the 10-byte segment of the program file in a table with 10 one-byte variables. The program in the repeater station would scan the received data to check whether all the 10 variables were non-empty, because incomplete segment could be received during the transmission. If the received data were non-empty, the repeater station would ask for the next segment of the program upgrade from the central station. Otherwise, the repeater station would proceed to send the program upgrade to the gateway station and send a signal to the CR1000 to acknowledge the complete reception of the program upgrade.

5.2.3.4 CR206 datalogger program at gateway station

Functions of the original CRBasic program for the CR206 datalogger at a gateway station was to send the sediment data received from the Stargate to the datalogger at the repeater station only. For OAP, the gateway station's datalogger also needed to receive the program upgrade from the CR206 at the repeater station and forward it to the Stargate at a gateway station through the RS232 serial port. The flowchart of the program is shown in Figure 5.8. The source code of the program is in the Appendix F.

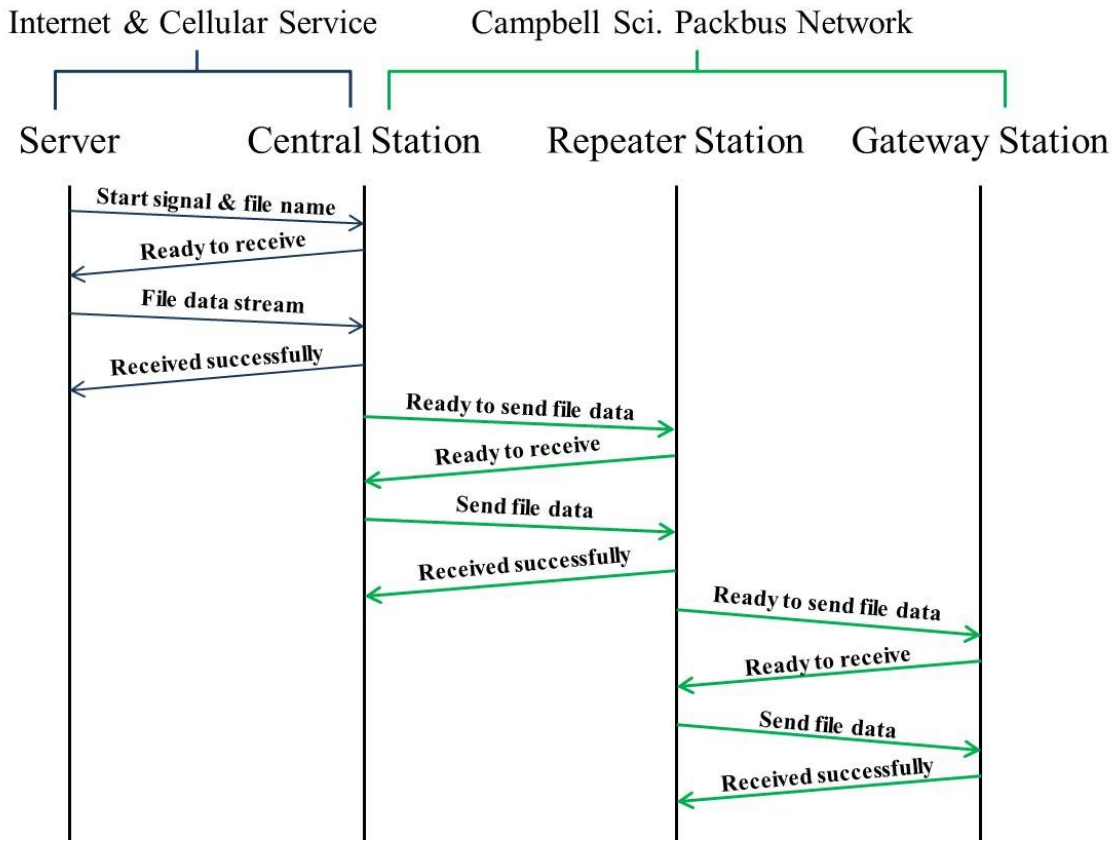
Like the program for the datalogger at the repeater station, the program for the gateway station would make sure that the sediment data was successfully transmitted at the beginning of each loop. During the next step, the program would check if the CR206 at the repeater station was ready to send a program upgrade. After receiving the 10 one-byte variables from the repeater station, the program in the gateway station would scan the received data to check whether all the 10 variables were non-empty and different from the data received previously in order to avoid saving redundant data. If the received data were non-empty nor not redundant, the gateway station would ask for the next segment of the program upgrade from the repeater station. Otherwise, the gateway station would proceed to forward the program upgrade to the Stargate through the RS232 serial port and send a signal to the CR206 at the repeater station to acknowledge the complete reception of the program upgrade.

Figure 5.8 Flow chart of CR206 datalogger program at gateway station



The wireless communications among the transceivers are summarized in Figure 5.9.

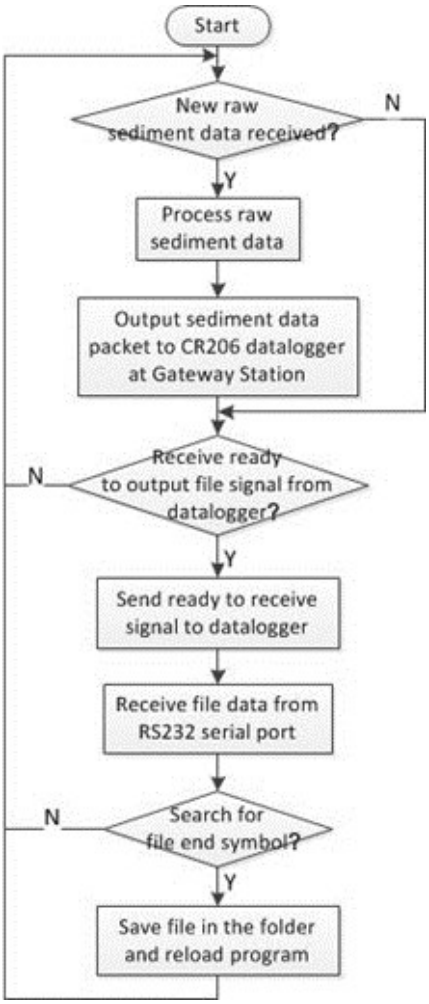
Figure 5.9 Communication among transceivers



5.2.3.5 Stargate program

In order to receive the data from the datalogger at the gateway station, the Stargate would continuously monitor the CR206 datalogger's RS232 serial port. After the sediment data was processed and sent to the datalogger, if the Stargate received a signal indicating that the data segment of the new program was ready, it would send an acknowledgement back to the datalogger to indicate that it was ready to receive the data segment. As the 10-byte segment was saved as 10 one-byte variables in the datalogger, they would also be delivered in 10 one-byte variables to the Stargate and saved in the Stargate's buffer. After the data was received, the Stargate would check if there was a predefined ending symbol among all the variables. Once the symbol was found, the Stargate would integrate all the variables in the buffer to a new program file, save it in the designated file folder, and call the system level command to reload the new program. The flowchart of the program is shown in Figure 5.10. The source code of the program is in the Appendix G.

Figure 5.10 Flow chart of Stargate program at gateway station



5.2.4 Deployment and installation

The OAP function was tested in the Manhattan – Fort Riley experimental site. A Raven XT cellular modem with AT&T I2Gold service was installed at the Colbert Hill central station to provide the dual-way communication. A CR1000 datalogger with a RF401 radio module was also installed at the central station. The program for the CR206 datalogger at the Cico Tank repeater station, the program for the CR206 datalogger at the Little Kitten Creek gateway station, and the program for the Stargate at the gateway station were all updated to the latest version to support OAP.

5.3 Results

5.3.1 File transmission in LRCN

File transmission for OAP was tested in the LRCN tier to evaluate the data transmission rate and the transmission error rate quantitatively. Program files in four different sizes were transmitted from the server computer to the remote CR1000 datalogger at the central station. Each file of the same size was transmitted five times. The transmission time was calculated using the difference between the time stamp the program file was copied to the predefined folder in the server computer and the time stamp logged in the CR1000 datalogger when the program file was saved. Table 5.1 shows the file sizes, the transmission times, and the average transmission time.

Data in the table show the trend that the average transmission time was basically determined by the file size. The data transmission rate was approximately 9.1 kbps. It was mainly limited by the baud rates of the cellular modem RS232 serial port and the CR1000 datalogger serial port, which were both set to 9.6 kbps.

Table 5.1 File size and transmission time of test trials in LRCN

File size (kB)	Transmission time (Second)					Average transmission time (Second)
	1st	2nd	3rd	4th	5th	
4.81	5	4	6	4	4	4.62
10.21	10	10	9	10	10	9.83
20.63	19	18	17	18	17	18.19
30.81	28	27	27	28	27	27.85

To check the data transmission error for the file transmission, the saved file after each trial was downloaded from the CR1000 datalogger and was compared with the original source program file in the server computer.

Through the file transmission tests, the new JAVA program running in the server computer and the CRBasic program running in the CR1000 datalogger at the central station were reliable to implement the OAP in the LRCN tier.

5.3.2 File transmission in MRWN

File transmission for OAP was also tested at the MRWN tier to evaluate the data transmission rate and the transmission error rate quantitatively. Program files in four different

sizes were transmitted from the server computer to the remote CR1000 datalogger at the central station, and the CR1000 would proceed to send the program files to the Stargate at a gateway station through two CR206 dataloggers at a repeater station and at the same gateway station respectively. Each file of the same size was transmitted five times. The transmission time was calculated using the difference between the time stamp logged in the CR1000 datalogger when the program file was saved and the time stamp logged in the Stargate when the program file was saved. Table 5.2 shows the file size, the transmission time, and the average transmission time.

Table 5.2 File size and transmission time of test trials in MRWN

File size (kB)	Transmission time (Second)					Average transmission time (Second)
	1st	2nd	3rd	4th	5th	
4.34	1640	1609	1597	1946	1643	1687
7.84	2983	2873	3280	3528	2904	3114
13.72	5152	6591	5863	4842	5435	5577
22.32	9866	8704	8068	10464	9867	9394

Data in the table show the trend that the average transmission time was basically determined by the file size. The data transmission rate was approximately 18.6 bps. The low data transmission rate was due to two reasons. First, the wireless transmission among the CR206 dataloggers at the repeater station and the gateway station was implemented by transmitting the variables predefined in the dataloggers' tables. It would be much slower than transmitting continuous data streams through the cellular service. Second, the CR206 datalogger only reserved 60 variables for programming. Only 10 variables could be used to save the new program file segment except for the variables used for sediment data transmission and program logic control. Therefore, the transmission frequency between the CR206 dataloggers was increased massively, hence causing long transmission period.

To check the data transmission error for the file transmission, the saved file after each trial was downloaded from the Stargate CF card and was compared with the original source program file in the server computer. All the files completely matched the original ones. Through the file transmission tests, the new programs running in the CR206 dataloggers at the repeater and gateway stations, and the program running in the Stargate at the gateway station could deliver the program upgrade correctly and completely within the MRWN tier.

5.4 Discussion

The program for the CR1000 datalogger at the central station was usually within 10 kB in length. With the data transmission rate used at the LRCN tier, the program file can be updated remotely in about 10 seconds using OAP. As mentioned in Section 5.1, the program for the datalogger at the central station stored the information on geographic distributions of the repeater and gateway stations. If the distributions changed, the program for the datalogger at the central station could be updated easily using OAP.

The program for the Stargate at the gateway station was usually more than 25 kB in length. Due to the low data transmission rate at the MRWN tier, the transmission would take more than three hours to upgrade the program. But if there is not a time constraint for remote program update, this method still can be used. A better way to implement OAP in MRWN tier is to replace the CR206 dataloggers with other wireless communication devices, which need to have a larger memory to buffer the data and to transmit the data in a streaming mode.

In addition, as OAP has enable the communication within the WSN from a one-way scheme to a bidirectional scheme, the network security needs to be considered in the future to avoid the system hacking. A firewall program can be deployed at the central station to protect the WSN system.

Chapter 6 - RADIO PROPAGATION IN DENSELY VEGETATIVE AREAS

6.1 Introduction

In the three-tier WSN, the three tiers covered different scales for wireless transmission with different technologies. The commercial cellular service was used in the LRCN tier for large scale wireless communication and was found the most reliable wireless communication technology used in the three-tier WSN. Installing the central stations within open areas further guaranteed the quality of signal transmission. The MRWN relayed the data through a moderately long distance (up to 16 km) with 915 MHz FHSS radios embedded in dataloggers. Because most paths between the transmitters and receivers in this tier were line-of-sight, signal losses due to blocking objects were minimal.

In the LWSN tier, all sensors were installed in streams. Thus, the sensor node, including the data acquisition system, the sensor control board, and the signal transmitter, could not be installed too far away from the stream. The signal transmitter was usually surrounded by grasses, bushes, and trees with dense foliage. The leaves, twigs, branches and trunks randomly distributed in the signal transmission path would absorb, diffract, and scatter the radio waves and, thus, significantly affect the quality of the received signal. As a result, even with a short range (within 100 m) between the transmitter and the receiver, signal attenuation could be significant. When the signal power was reduced to a certain level, the receiver would no longer sense the signal, thus causing complete data loss. Because of these problems, a study on radio propagation in densely vegetative areas was needed.

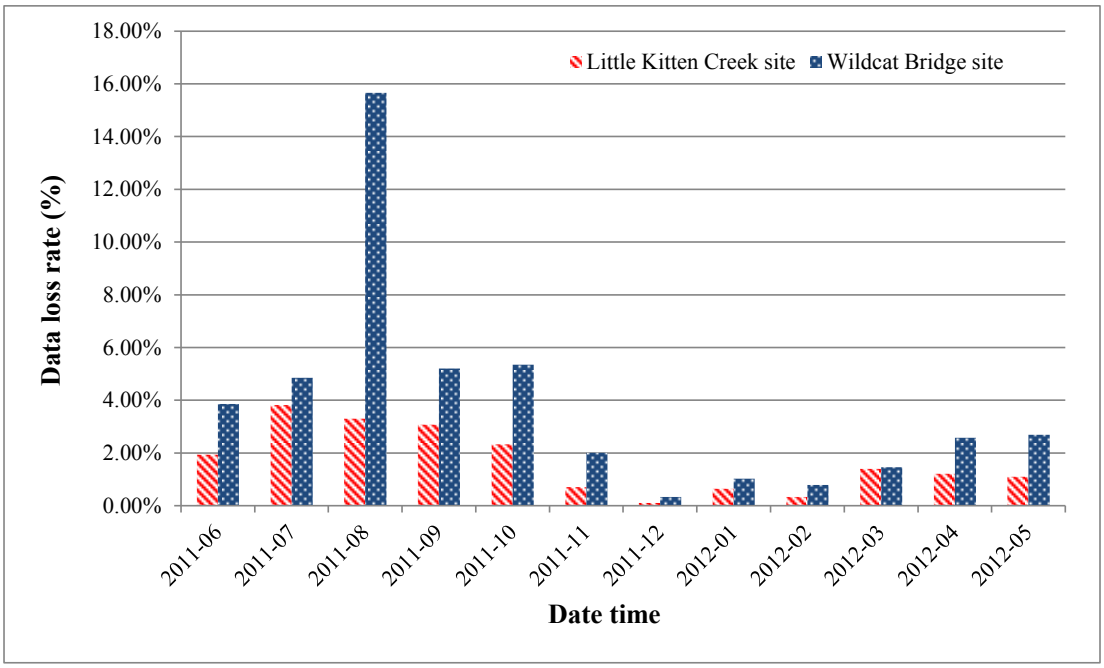
A previous research (Han, 2011) qualitatively analyzed the radio propagation in the dense vegetation environment within the LWSN tier based on three basic propagation mechanisms - reflection, diffraction, and scattering (Rappaport, 2002). It was demonstrated that, by carefully selecting the locations of the transmitter and receiver, reflections on the radio wave path could be reduced. To create a large clearance volume in the first Fresnel zone (Rappaport, 2002), antenna towers should be built to specific heights. The previous study also found that a high carrier frequency produced a shorter wavelength to abate the scattering effect. These technologies all

helped to improve the signal quality. However, the previous research did not study signal attenuation due to foliage and vegetation quantitatively.

Radio propagation models and path loss models have been used to estimate the power gain between a transmitter and a receiver. However, it is usually difficult to find the parameters needed for the models by direct measurements, especially within a complicated environment. Very often, an empirical approach based on curve fitting is used to create propagation models. This approach has the advantage of implicitly taking into account all propagation factors, both known and unknown, through actual field measurements. Although many empirical models have been derived to analyze the path loss in the outdoor environment, some even directly investigated the impact of plants on radio propagation, these models cannot be directly used to estimate the radio propagation in the LWSN tier for this study. Firstly, the transmitter-receiver (T-R) separation in the LWSN tier was very short (usually within 50 m), while most radio propagation models dealt with large T-R separations (up to several kilometers). The signal attenuation features in such a short range may not be presented appropriately by those models. Secondly, the antenna height in the LWSN tier was much lower (usually under 2 m) than those used in the models. Usually, increasing the antenna height would reduce the energy loss caused by ground reflection. However, installing tall antennas near the sensor nodes was difficult. Finally, the output power of the transmitter used in the LWSN tier was very small (only 1 mW). The path loss range only occupied a small portion of that studied by most path loss models, which were derived from systems with high power transmitters. Thus, it was necessary to develop a path loss model that fits the conditions of the LWSN tier in this study.

Figure 6.1 shows the data loss rate measured within a one-year period in the LWSN tier at the Little Kitten Creek and Wildcat Creek experiment sites. It was clear that the data loss rates measured at the two sites had a similar trend - higher in summer months and lower in winter months. This is quite understandable because vegetation is usually denser in summer. As the transmitter and receiver were installed close to creeks, variations in air temperature and relative humidity might also impact the signal strength. These effects also needed to be studied.

Figure 6.1 History of data loss rate at Little Kitten Creek and Wildcat Bridge sites



Thus, the main purpose of this research was to understand the radio propagation in densely vegetative areas, to improve the quality of wireless transmission in these areas, and to establish guidelines for installation of sensor nodes and gateway stations in the LWSN tier. Specific objectives included:

(1) To develop a path loss model for low power wireless signal transmission with short T-R separation in densely vegetative areas;

(2) To investigate signal strength variation due to changes in air temperature and relative humidity.

6.2 Methodology

6.2.1 Path loss measurement and data logging

Signal transmission in the LWSN tier was achieved by two wireless Motes (MICAz, MEMSIC Inc., Andover, MA): one as a transmitter, attached to the sensor control board; and the other as a receiver, attached to the Stargate. The Motes, as shown in Figure 6.2, used a 2.4 GHz, direct sequence spread spectrum radio for communication.

Figure 6.2 Top view of a MICAz mote

The RF power of the transmitter mote (P_t) was set to the maximum value of 0 dBm (1 mW). The power sensitivity of the receiver mote was -94 dBm. For each received message, the receiver mote measured the received signal strength indicator (RSSI) value and attached it to the raw data. The Stargate program then separated the RSSI value from the raw data. The received power (P_r) in dBm can be calculated by Equation 6.1:

$$P_r[dBm] = RSSI - 45 \qquad (6.1)$$

To reduce the signal attenuation by vegetation, high-gain antennas were used at both the transmitter and the receiver ends. In the original setting, the Yagi directional antennas were used at both ends, because the locations of the transmitter and receiver were fixed. However, for the experiment of path loss measurement, because the locations were no longer fixed, a pair of omni-directional antennas were used. The gain of the antenna was 8.5 dBi.

The total path loss could be calculated by Equation 6.2.

$$PL = (P_t + G_t) - (P_r - G_r) \qquad (6.2)$$

where:

P_t is the transmitter power in dBm;

P_r is the received power in dBm and can be calculated by Equation 6.1;

G_t is the transmitter's antenna gain in dBi;

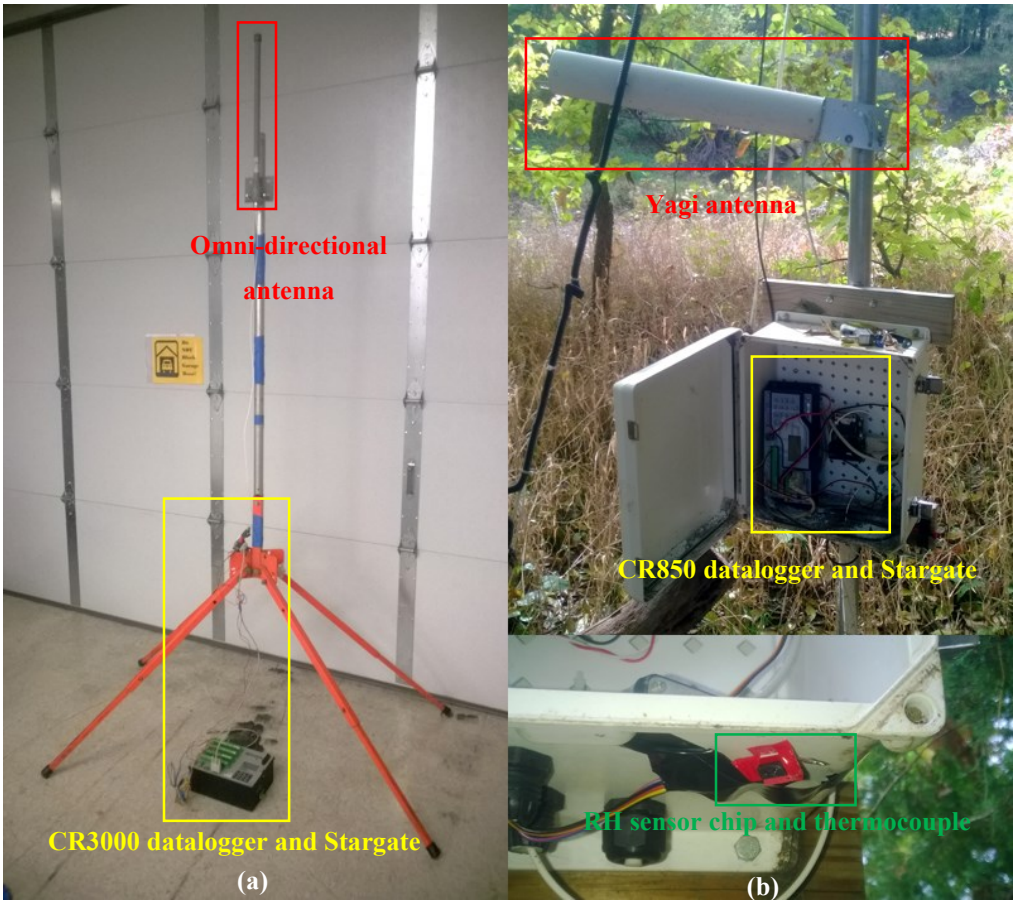
G_r is the receiver's antenna gain in dBi;

PL is the path loss in dB.

To measure the path losses at variable T-R separations, a portable Gateway Station was built, as shown in Figure 6.3(a). An Omni-directional antenna was mounted on the top of a

height-adjustable tripod and was connected to a MICAz mote on a Stargate. A CR3000 datalogger with an internal 12 VDC battery powered all electronic devices and read the RSSI values from the Stargate's serial port. For in-field, real-time measurement of RSSI, the sampling rate was 0.33 Hz.

Figure 6.3 Two types of gateway station to record RSSI value
(a) the portable gateway station; (b) the stationary gateway station.



To investigate the impact of air temperature and relative humidity (RH) on radio propagation, the receiver's RSSI value was measured at the original settings (Figure 6.3 (b)). A CR850 datalogger sampled the RSSI value every three minutes. A RH sensor (Honeywell HIH-4030) and a thermocouple wire were connected to the CR850 datalogger. The air temperature and RH value were recorded at the same time the RSSI value was recorded.

6.2.2 Path loss in densely vegetative area

The path loss between the transmitter and the receiver in densely vegetative area could be considered as the summation of the path loss in the open area and the path loss due to the attenuation by dense vegetation, as expressed in Equation 6.3.

$$PL = PL_{Open} + PL_{Vegetation} \quad (6.3)$$

Both theoretical and measurement-based propagation models have indicated that the average received signal power in an open area decreases logarithmically with the distance (Hata, 1980). Thus, the log-distance path loss model was used to estimate the open area path loss for an arbitrary T-R separation with the 2.4 GHz MICAz motes, as expressed in Equation 6.4 (Rappaport, 2002).

$$PL_{Open}(d) = PL_0 + 10nlog_{10}(d) \quad (6.4)$$

where

PL_{Open} is the average path loss in an open area in dB;

PL_0 is a constant power gain in dB;

n is the path loss exponent;

d is the T-R separation distance in meters.

The path loss exponent n indicates the rate at which the path loss increases with distance. When plotted on a log-scale for the distance, the modeled path loss is a straight line with a slope of 10n dB per decade. The value of n depends on the specific propagation environment. The classic free space path loss model is expressed with n equal to 2. When obstructions are present, n will have a larger value.

For the radio propagation due to the presence of vegetation, the exponential decay model, which was first proposed by Weissberger (Weissberger 1981), was commonly used (CCIR 1986; Meng et al. 2009). The general form was shown as Equation 6.5.

$$PL_{Vegetation} = A \times f^B \times dp^C \quad (6.5)$$

where

$PL_{Vegetation}$ is the path loss due to the foliage in dB;

f is the radio wave carrier frequency in GHz;

dp is the foliage depth between the transmitter and the receiver in meters;

A, B and C are the derived parameters through regression analysis.

The model is applicable for point-to-point communication links, which have obstructions by foliage. The major advantage of the exponential decay model is its simplicity, as it does not require the knowledge on the geometry of the foliage.

With Equation 6.4 and Equation 6.5 substituted into Equation 6.3, the total path loss can be expressed as Equation 6.6.

$$PL = PL_0 + 10nlog_{10}(d) + A \times f^B \times dp^C \qquad (6.6)$$

In order to derive the parameters *n*, *A*, *B*, and *C* in Equation 6.6, three experiments were conducted.

6.2.2.1 Path loss in an open area with variable T-R separations

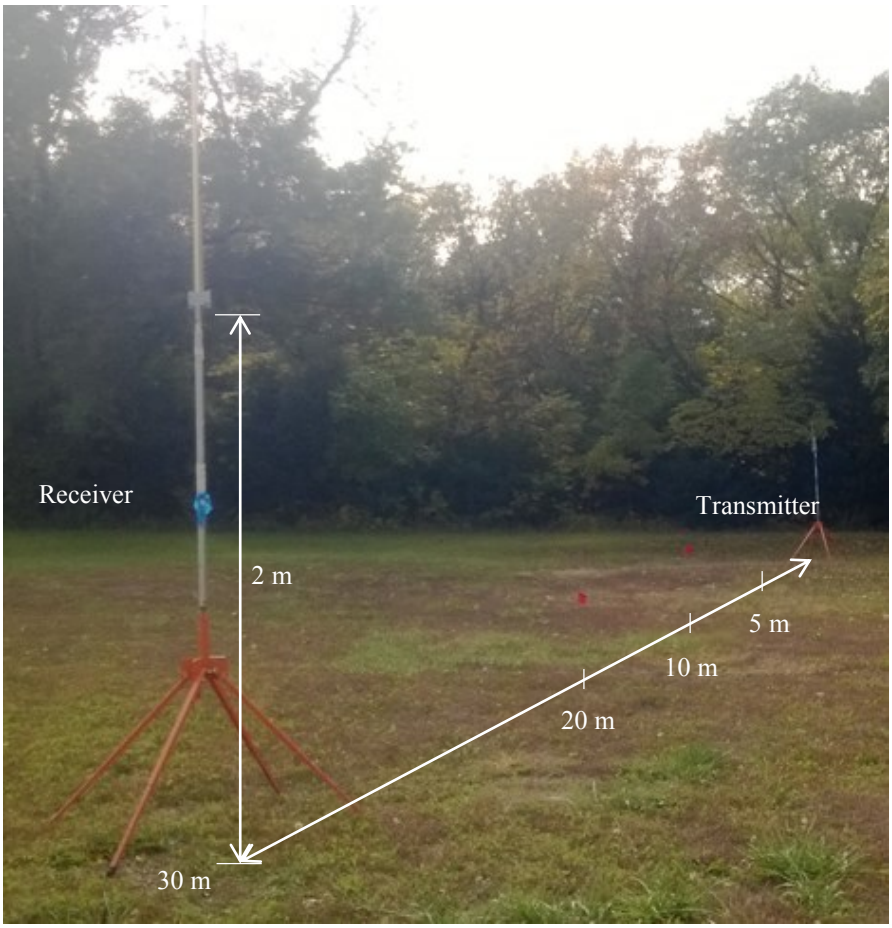
The purpose of this experiment was to obtain the path loss exponent *n* in Equation 6.6. As the path losses were measured in an open area, the path losses due to the vegetation were ignored, thus changing Equation 6.6 to

$$PL = PL_0 + 10nlog_{10}(d) \qquad (6.7)$$

The path loss exponent *n* was derived using a regression analysis by using path loss values (*PL*) measured at different T-R separations (*d*).

The experiment was conducted in an open area close to the Little Kitten Creek sensor site, as shown in Figure 6.4. Two Omni-directional antennas were adjusted to the same height and installed on two quad-pods. The antenna height was set to 2m above the ground. This height was selected because it could be used in both the open-area and the densely vegetative area experiments. Excessive heights would cause significant blockage by foliage and braches of the trees. On the other hand, excessively low antenna installation would cause signal attenuation by absorption of ground and grasses.

Figure 6.4 Path loss measurement in the open area



During the experiment, the transmitter was placed at a fixed location and was continuously sending pseudo-random raw data to the receiver. The mote at the receiver measured the RSSI value and forwarded it to a CR3000 datalogger, where the RSSI values were saved. The receiver was moved so that the separation increased from 5m to 10m, and from 10m to 50m at 10m increments. At each separation, the receiver stayed stationary for more than five minutes. As the datalogger recorded an RSSI value every three seconds, more than 100 RSSI values were collected during this time period. The average RSSI values were used in the regression analysis.

6.2.2.2 Path loss due to variable foliage depth with a constant T-R separation

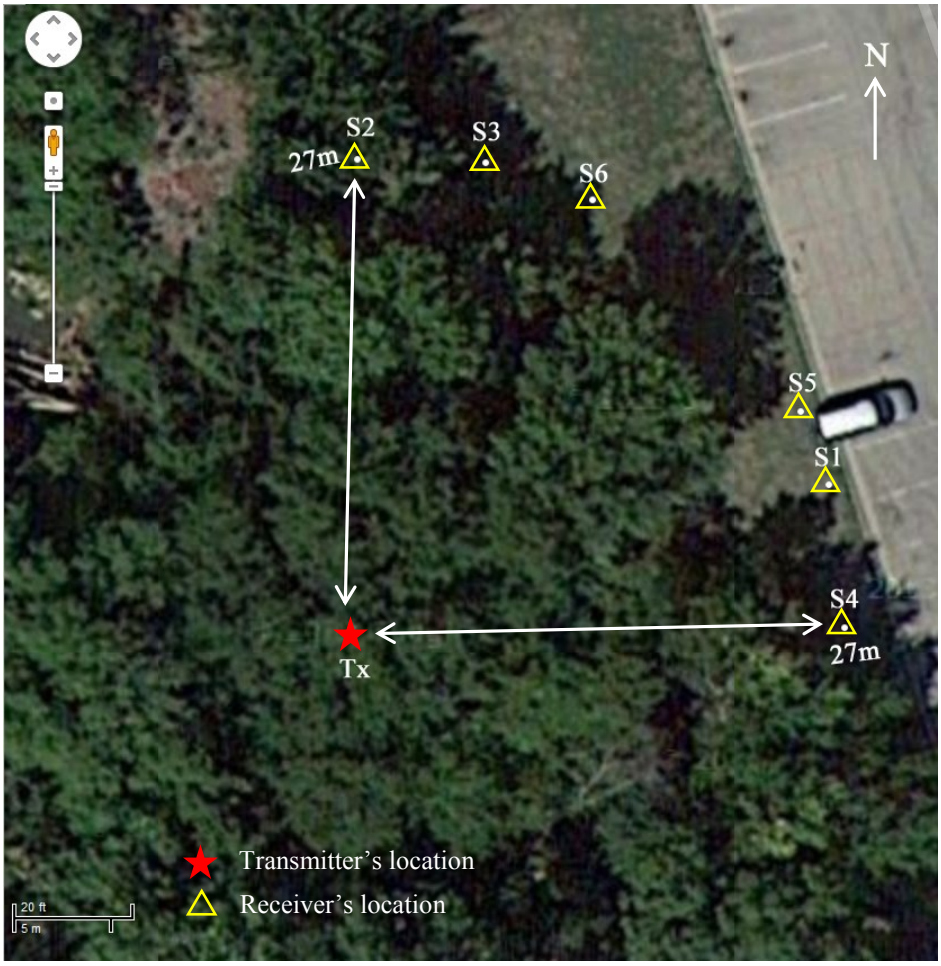
The objective of the second experiment was to derive the parameters A, B, and C in Equation 6.6. As the T-R separation was set to a constant distance in this experiment, PL_{Open} was constant, which was derived using Equation 6.7. In addition, as the radio carrier frequency was also constant at 2.4 GHz, Equation 6.6 was simplified to

$$PL = PL_{Open} + \alpha \times dp^{C} \qquad (6.8)$$

Parameters α and c were derived through a regression analysis using path loss values (PL) measured at different vegetative depths (dp) between the transmitter and the receiver.

The second experiment was also conducted near the Little Kitten Creek sensor site using the same transmitter and the receiver. In Figure 6.5, the red star indicates the transmitter's location, and the yellow triangles indicate the receiver's locations. The T-R separation was selected as 27m, because the signal became too weak to receive beyond this distance.

Figure 6.5 Locations of the transmitter and the receiver at a constant T-R separation



To quantify the foliage depth (dp) for each receiver location, the length of each "foliage block", which was defined as densely grown leaves, twigs, and trunks, was measured. Because it was difficult to measure the density of the foliage blocks of leaves and twigs and to describe the distribution of leaves and twigs within a block, the foliage depth was measured as the distance between two edge points of the foliage block present along the straight line between the transmitter and the receiver using a tape measure. Diameters of these trunks were calculated

from the measurement of their circumferences. The total foliage depth was the sum of the foliage depths of all the foliage blocks.

Six receiver locations were selected to measure the path losses due to variable vegetative depths with a constant T-R separation. Figure 6.6 to Figure 6.11 give the views of the transmission path from the transmitter and the receiver for these locations. The rectangle in red dashed lines shows the location of the transmitter and the receiver.

Figure 6.6 Views from (a) the transmitter and (b) the receiver for receiver location S1

(a) **(b)**

When the receiver was placed at S1, as shown in Figure 6.6, the transmitter could be seen from the receiver side, as there were only some twigs and very little leaves present along the propagation path. The measured foliage depth was 1.00m.

Figure 6.7 Views from (a) the transmitter and (b) the receiver for receiver location S2

(a) **(b)**

When the receiver was placed at S2, as shown in Figure 6.7, the transmitter (receiver) could not be seen due to foliage. Two trees with trunk circumferences of 0.20m and 1.00m were located in the propagation path. The measured foliage depth was 12.00m.

Figure 6.8 Views from (a) the transmitter and (b) the receiver for receiver location S3

(a) **(b)**

When the receiver was placed at S3, as shown in Figure 6.8, the transmitter and receiver still could not be seen from each other. Three trees with trunk circumferences of 1.60m, 0.14m, and 2.00m were located in the propagation path. The measured foliage depth was 6.40m.

Figure 6.9 Views from (a) the transmitter and (b) the receiver for receiver location S4

(a) **(b)**

When the receiver was placed at S4, as shown in Figure 6.9, the transmission path was still not line-of-sight. There were two small trees with circumferences of 0.11m and 0.21m in the propagation path. The measured foliage depth was 5.10m.

Figure 6.10 Views from (a) the transmitter and (b) the receiver for receiver location S5

(a) **(b)**

When the receiver was placed at S5, as shown in Figure 6.10, a huge block of bushes was present in the radio propagation path. The measured foliage depth was 10.70m.

Figure 6.11 Views from (a) the transmitter and (b) the receiver for receiver location S6

(a) **(b)**

When the receiver was placed at S6, as shown in Figure 6.11, the transmitter was blocked by a tree with a trunk circumference of 1.00m. The measured foliage depth was 6.30m.

Table 6.1 Summary of foliage depths for the second experiment

Receiver's location	S1	S2	S3	S4	S5	S6
T-R separation (m)	27.00	27.00	27.00	27.00	27.00	27.00
Total foliage depth (m)	1.00	12.38	7.59	5.10	10.70	6.62

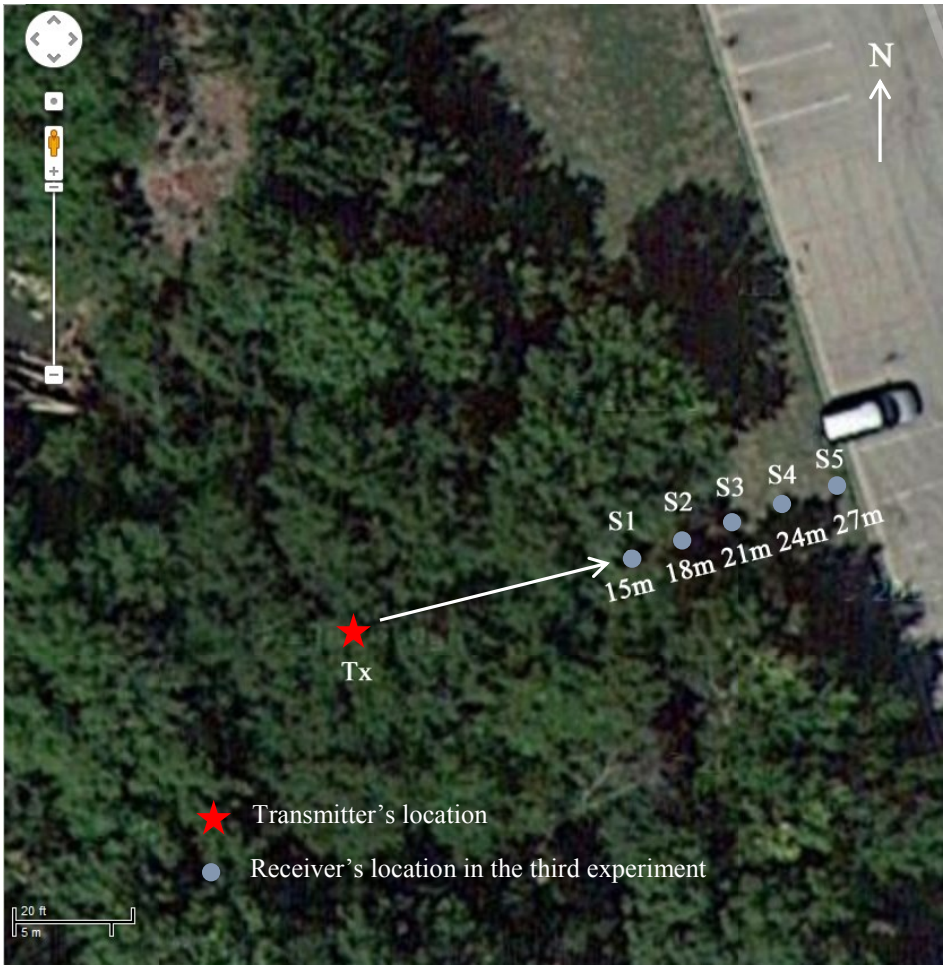
At each location, the receiver stayed stationary for more than five minutes. Thus, more than 100 RSSI values were collected. The path losses at six receiver's locations were calculated based on the average RSSI values. The fitting parameters in the exponential decay model were derived through a regression analysis between the path losses and the foliage depths.

6.2.2.3 Path loss due to a constant foliage depth with variable T-R separations

After the first and the second experiment, the path loss model was derived. The purpose of the third experiment was to compare the real path loss measurements with the path loss values predicted by the path loss model.

The blue dots in Figure 6.12 indicate the receiver's locations in the third experiment. The receiver's RSSI was measured at different T-R separations, ranging from 15m to 27m with 3m increments.

Figure 6.12 Locations of the transmitter and the receiver at variable T-R separations



The foliage depths were measured using the same method as that used in the second experiment. As there was no vegetation between location S1 and location S5, the foliage depth did not change, as shown in Table 6.2.

Table 6.2 Summary of foliage depths for the third experiment

Receiver's location	S1	S2	S3	S4	S5
T-R separation (m)	15.00	18.00	21.00	24.00	27.00
Total foliage depth (m)	1.00	1.00	1.00	1.00	1.00

At each location, the receiver stayed stationary for more than five minutes. Thus, more than 100 RSSI values were collected to obtain an average path loss value.

6.2.3 Variation in signal strength due to air temperature and relative humidity

To investigate the variation in signal strength due to changes in air temperature and relative humidity, the receiver's RSSI value, the air temperature, and the relative humidity were measured at the same time. The measurements were taken continuously at Little Kitten Creek site and Wildcat Bridge site in Manhattan, KS, from October 7 to 23, 2013. Through calculating the correlation coefficients between signal strength and air temperature, and between signal strength and relative humidity, the effect of air temperature and relative humidity on signal strength were analyzed.

6.3 Results

6.3.1 Path loss in an open area with variable T-R separations

Using the RSSI values measured at variable T-R separations in the open area, average path losses were calculated, as shown in Table 6.3.

Table 6.3 Path losses at variable T-R separations in the open area

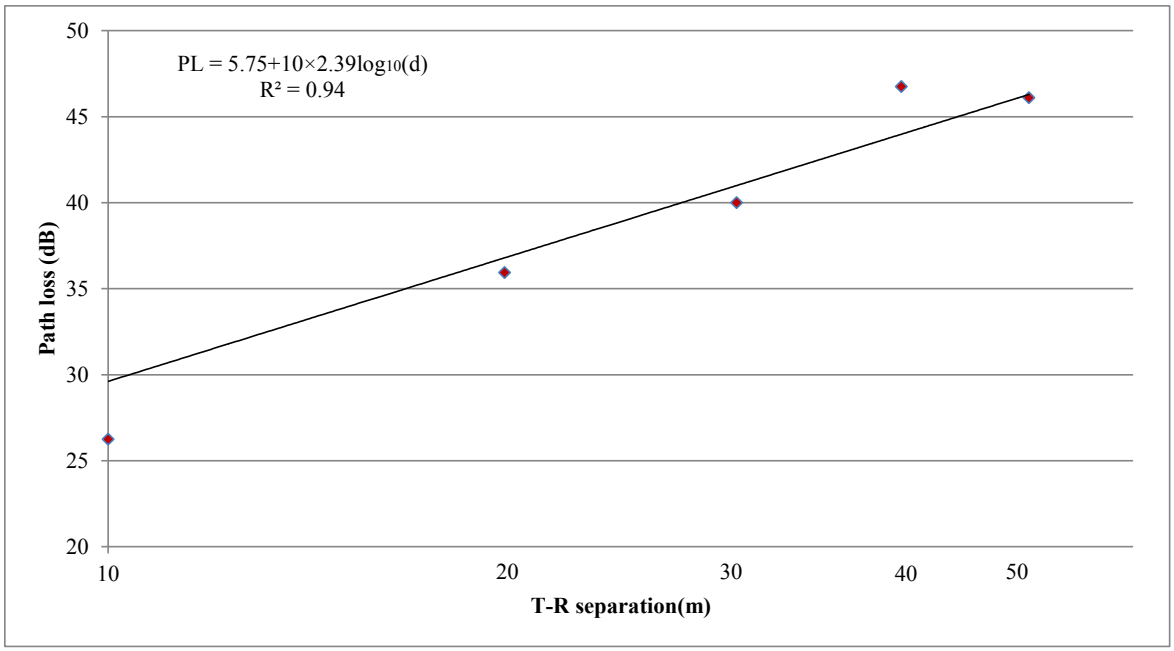
T-R separation (m)	5	10	20	30	40	50
Path loss (dB)	25.08	26.25	35.93	40.00	46.75	46.10

The path loss exponent *n* was derived through a linear regression, as shown in Figure 6.13. The R^2 value was 0.94,

$$PL(d) = 5.75 + 10 \times 2.39 log_{10}(d) \qquad (6.9)$$

As shown in Equation 6.9, the path loss exponent *n* was equal to 2.39, which was larger than 2 - the path loss exponent value for free space. The extra attenuation was probably caused by signal power loss due to the limited antenna gain and height.

Figure 6.13 Path losses in the open area



6.3.2 Path loss model derived from the path loss due to variable foliage depth with a constant T-R separation

Average path loss values were calculated using RSSI values measured at six different receiver's locations in the second experiment, as shown in Table 6.4.

Table 6.4 Path loss values at six locations due to variable foliage depth

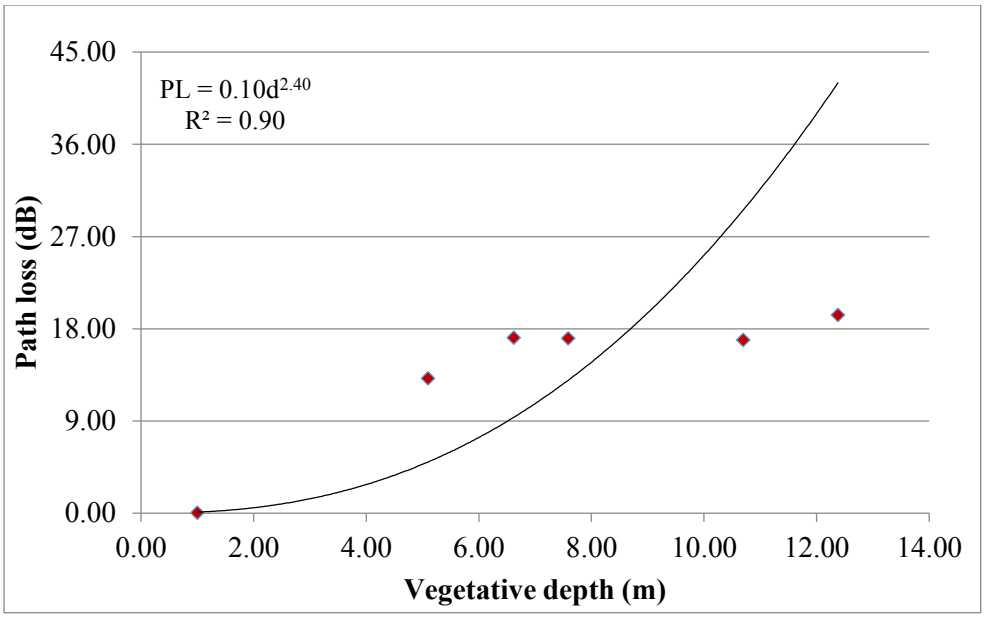
Receiver's location	S1	S2	S3	S4	S5	S6
Foliage depth (m)	1.00	12.38	7.59	5.10	10.70	6.62
Measured path loss (dB)	40.02	59.31	57.05	53.13	56.88	57.09
Path loss in open area (dB)	39.96	39.96	39.96	39.96	39.96	39.96
Excessive path loss (dB)	0.06	19.35	17.09	13.17	16.92	17.13

As the T-R separation in the second experiment was constant, the path loss in the open area was the same. The excessive path losses were the path losses caused by the vegetation in the radio propagation path.

The parameters (α and c in Equation 6.8) in the exponential decay model was fitted by regression analysis, as shown in Appendix H. The path loss model was shown in Equation 6.10.

$$PL(dp) = 0.10dp^{2.40} \qquad (6.10)$$

Figure 6.14 Path losses due to variable foliage depths



Thus, the total path losses between the transmitter and the receiver was the sum of the path loss in the open area, which was a function of the T-R separation distance "*d*", and the path loss due to vegetation, which was the function of foliage depth "*dp*", as expressed in Equation 6.11.

$$PL(d, d_p) = 5.75 + 10 \times 2.39 log_{10}(d) + 0.10 dp^{2.40} \quad (6.11)$$

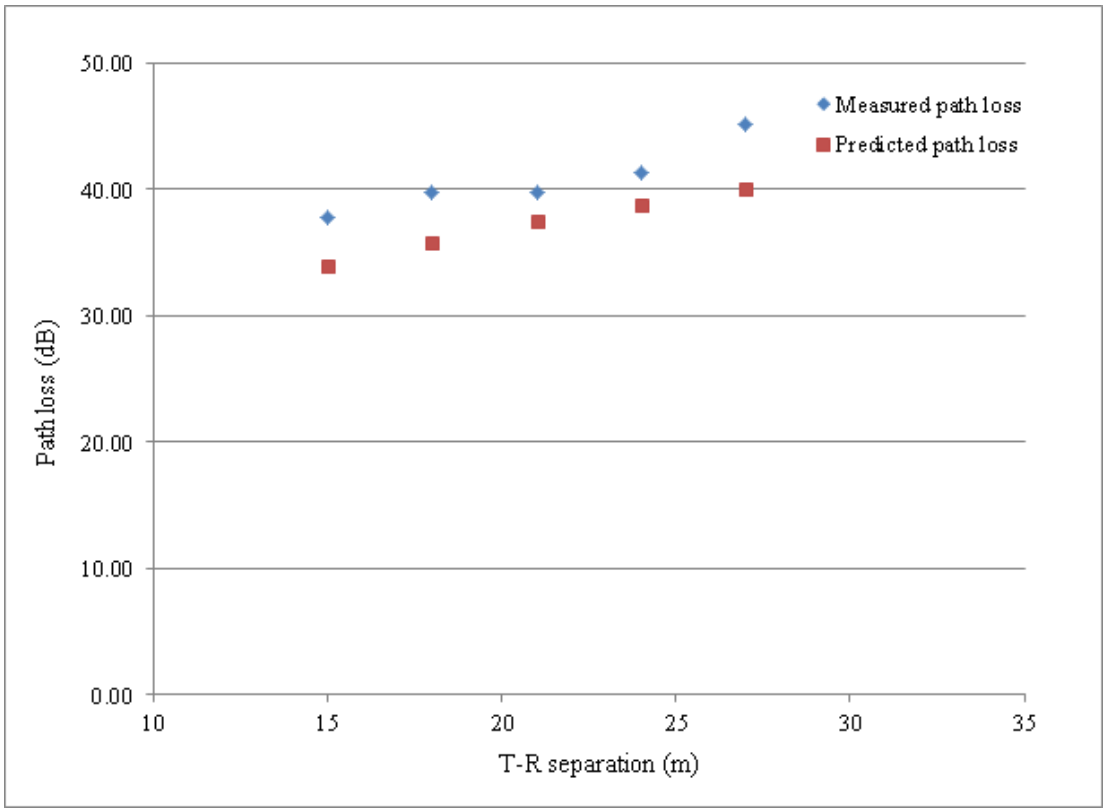
6.3.3 Comparison between the measured and predicted path losses in the experiment of constant foliage depth with variable T-R separation distances

Average path loss values were calculated based on the RSSI values measured in the experiment of constant foliage depth with variable T-R separation distances, as shown in Table 6.5. The predicted path loss values were calculated using Equation 6.11, also shown in Table 6.5.

Table 6.5 Path loss values due to constant foliage depth with variable T-R separation distances

T-R separation (m)	15	18	21	24	27
Foliage depth (m)	1.00	1.00	1.00	1.00	1.00
Measured path loss (dB)	37.82	39.75	39.71	41.38	45.10
Predicted path loss (dB)	33.96	35.85	37.45	38.84	40.06

Figure 6.15 Comparison between the measured and predicted path loss values



The root mean square value of errors between the predicted and the measured path loss values was 3.66dB. In this experiment, the path loss values predicted using Equation 6.11 were relatively close to the measured path loss values, when there was small depth (around 1.00m) of foliage between the transmitter and the receiver. To evaluate the prediction errors when there are larger foliage depth, further experiment needs to be conducted. It is also noticed, as shown in Figure 6.15, the predicted path loss values were always smaller than the measured path loss values. This was probably because that the measured vegetation used to compute the predicted path loss was smaller than the actual vegetation that attenuated the signal strength. The measured foliage depth was only considered as the length of the vegetation existing along the line that connected the transmitter and the receiver. But the actual vegetation should be considered to exist in the Fresnel zone between the transmitter and the receiver.

6.3.4 Variation in signal strength due to air temperature and relative humidity

Figure 6.16 and Figure 6.17 show path loss (in red color), air temperature (in dark blue color), and relative humidity (in green color), measured from October 9 to 16, 2013 at the Little Kitten Creek and Wildcat Bridge sites, respectively.

Figure 6.16 Path loss, air temperature, and relative humidity measured at Little Kitten Creek site, from October 9 to 16, 2013

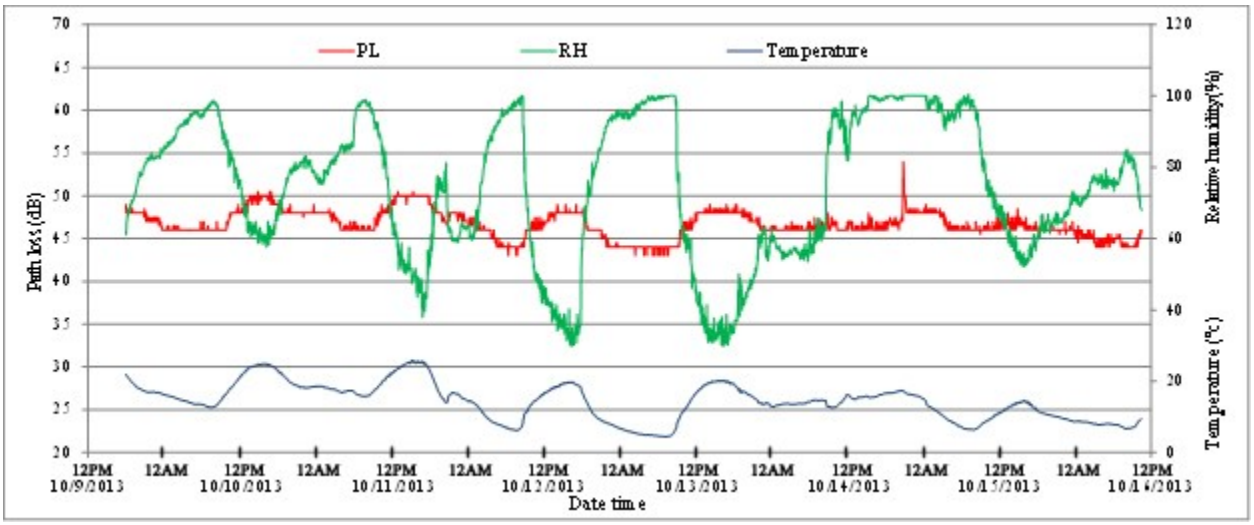
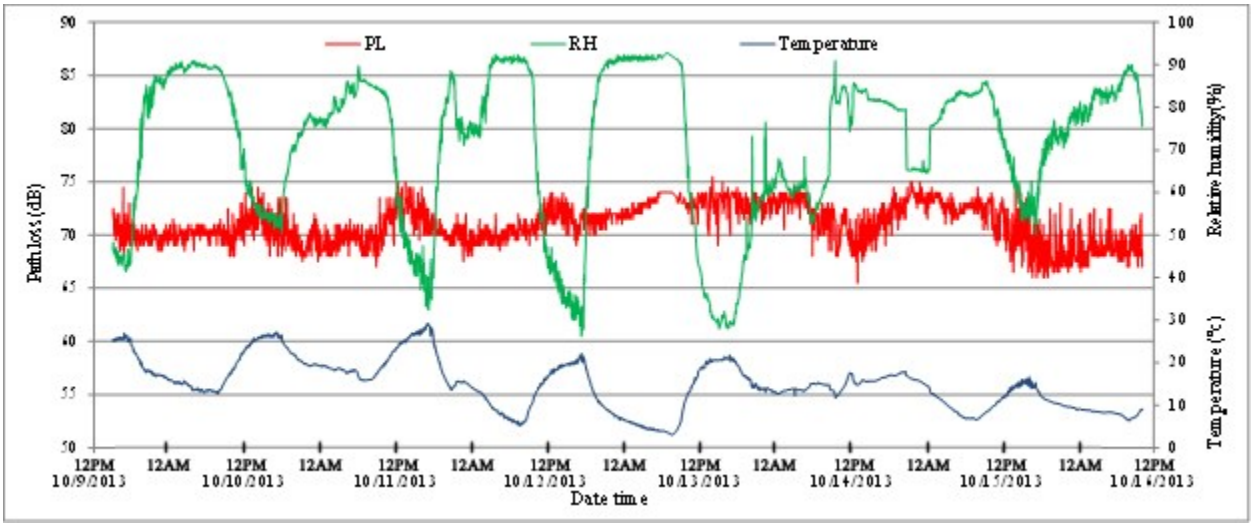


Figure 6.17 Path loss, air temperature and relative humidity, at Wildcat Bridge site, from October 9 to 16, 2013



From Figure 6.16, there may exist a correlated relationship between the path loss and the air temperature at Little Kitten Creek site. It was difficult to see any clear correlated relationship between the path loss and the relative humidity. From Figure 6.16 and Figure 6.17, it was observed that the average path loss measured at Little Kitten Creek site was much less than that at Wildcat Bridge site. It was probably because that the vegetation at Wildcat Bridge site was much denser (Figure 6.18).

Figure 6.18 Views from the transmitter at Little Kitten Creek site and Wildcat Bridge site

Linear correlation coefficients between path loss and air temperature and between path loss and relative humidity, as measured from October 7 to 23, 2013, are shown in Table 6.6.

Table 6.6 Linear correlation coefficient values

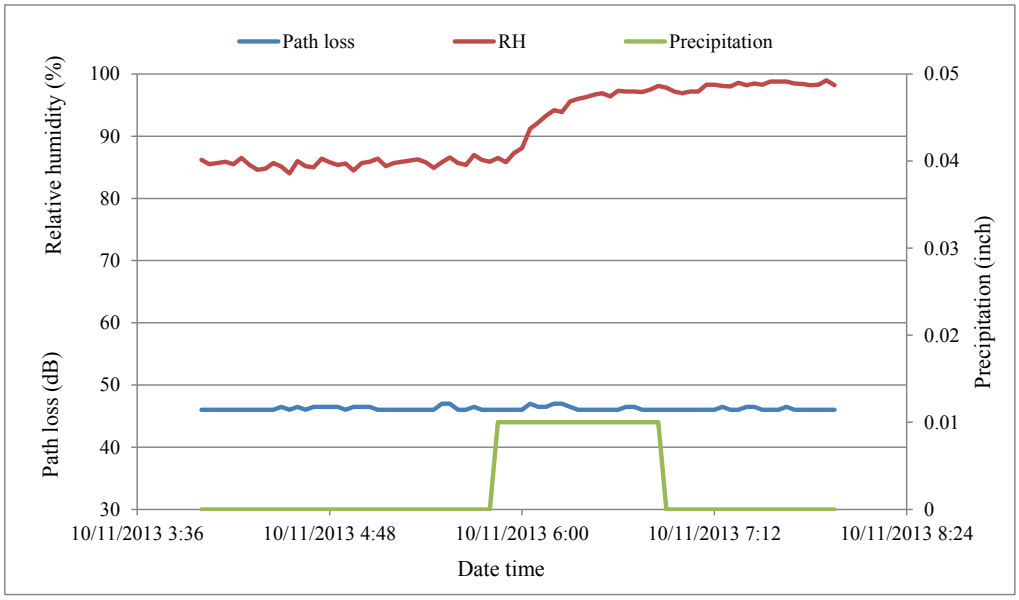
Location	Linear correlation coefficient	
	Path loss with temperature	Path loss with RH
Little Kitten Creek site	0.68	-0.44
Wildcat Bridge site	0.08	-0.18

From the values, there existed a moderate linear relationship between path loss and temperature and between path loss and relative humidity at Little Kitten Creek site. There was no apparent linear relationship between path loss and temperature and a very weak linear relationship between path loss and relative humidity at Wildcat Bridge site. Thus, the temperature and the relative humidity could be variables to affect the path loss at Little Kitten

Creek site. But there was not enough evidence to indicate that the temperature and the relative humidity substantially influenced the path loss at Wildcat Bridge site.

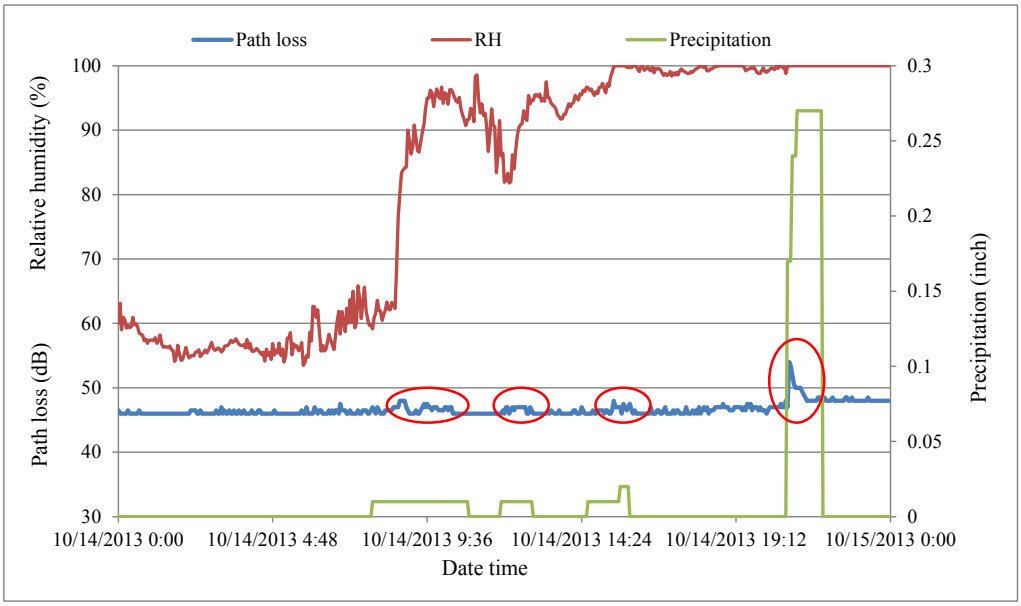
During the data logging period, two thunderstorm events occurred. Figure 6.19 and Figure 6.20 show the path loss, the relative humidity, and the accumulative precipitation during the thunderstorm events.

Figure 6.19 Path loss, relative humidity, and accumulative precipitation during the thunderstorm on October 11, 2013 at Little Kitten Creek site



From Figure 6.19, no obvious changes in path loss can be observed during the rainfall period.

Figure 6.20 Path loss, relative humidity, and accumulative precipitation during the thunderstorm on October 14, 2013 at Little Kitten Creek site



However, from Figure 6.20, four obvious variations in path loss can be observed. The path loss increased almost 10dB at the beginning of the last severe rainfall event. One possible explanation is the obstructions caused by downfall between the transmitter and the receiver. Lightening may be another possible cause.

6.4 Discussion

Due to the experiment design to derive this path loss model, there are some restrictions to use this model. Firstly, the path loss model in the open area was derived at a constant T-R separation distance of 50m. It is doubtful that the model would give accurate prediction when the T-R separation is beyond 50m. Secondly, the exponential decay model was established using foliage depths of less than 14m. The prediction errors may become large for longer foliage depths between the transmitter and the receiver. Additionally, the model can be only used for the radio frequency of 2.4GHz. Therefore, a possible use of this path loss model is to assist in the selection of the transmitter and the receiver locations, when the same types of radio devices are used.

Although this model may not predict the path loss as accurate as the deterministic model, the main advantage of using this model is its simplicity. The path loss could be predicted by

simply measuring the T-R separation distance and the foliage depth between the transmitter and the receiver, rather than to determine numerous variables in the physical laws.

In this research, we found that high air temperature and heavy rainfall could also become factors that impact the signal strength in LWSN. This may explain the lower data loss rate we have observed during the winter season, as shown in Figure 6.1. In the future research, an ideal data collection period would be a whole year in order to observe the variation in signal strength during four seasons.

Chapter 7 - CONCLUSION

In this research, the wireless transmission performance of a three-tier WSN was improved, and the radio propagation was investigated in the densely vegetative environment. The following conclusions can be drawn from this study:

1. The interval between the generation of the raw data and the final display of the SSC value was shortened by reducing the Stargate buffer size and the data packet length.

2. The data loss rate was reduced after applying the semaphore mechanism to identify the new raw data received by the Stargate, and by sending the acknowledgement signal to the transmitter once the data packet was received by the receiver.

3. The data transmission throughput was enlarged by adopting a time and event driven scheduling method, hence achieving to transmit all the raw velocity data back to the database server.

4. A central station using MBC technology was developed and deployed in Manhattan – Fort Riley experimental site. After seven months field experiments, the average data loss rate of MBC network was 1.09% and the average data transmission error rate was 4.41%. The average transmission time for a SSC data packet was within five minutes.

5. The MBC technology can be used as a long distance communication method. But due to the unstable availability of the meteor trails, the MBC system can be only used as a backup tool to the cellular service to transmit a small amount of information.

6. By implementing OAP in the WSN, the programs running in the central station could be updated remotely with a data transmission rate of 9.1 kbps, but the programs in the gateway station could be updated with a data transmission rate of only 18.6 bps. To update the program in MRWN tier more rapidly, wireless communication devices, which have a larger memory to buffer the data and can transmit the data in a streaming mode, are needed.

7. A radio propagation model was derived to predict the path loss in a densely vegetative area, with restrictions to a 50m T-R separation distance, a 14m foliage depth, and a 2.4GHz radio carrier frequency. The main advantage of this model is its simplicity to predict the path loss.

8. In addition to the attenuation by vegetation, signal strength in the LWSN tier could also be influenced by ambient air temperature, relative humidity, and heavy rainfall.

References

Adly, I., H. F. Ragai, A. El-Hennawy and K. A. Shehata. 2010. Over-The-Air Programming of PSoC sensor interface in wireless sensor networks. In 2010 15th IEEE Mediterranean Electrotechnical Conference, 997-1002.

Al-Nuaimi, M. O. and A. M. Hammoudeh. 1994. Measurements and Predictions of Attenuation and Scatter of Microwave Signals by Trees. IEE Proceeding on Antennas and Propagation 141(2).

Antonio-Javier, G., G. Felipe and G. Joan. 2011. Wireless sensor network deployment for integrating video-surveillance and data-monitoring in precision agriculture over distributed crops. Computers and Electronics in Agriculture 75(2): 288-303.

Aschenbruck, N., J. Bauer, J. Bieling, A. Bothe and M. Schwamborn. 2012. Selective and Secure Over-The-Air Programming for Wireless Sensor Networks. In 21st International Conference on Computer Communications and Networks, 1-6.

Baker, S. P. 2010. Analyzing the Impacts of Tree Canopy on Cellular Radio Networks. PhD diss. Greensboro: University of North Carolina.

Brown, D. 1985. A Physical Meteor Burst Propagation Model and Some Significant Results for Communications and Systems Design. IEEE Journal on Selected Areas in Communications 3(5): 745-755.

Burrell, J., T. Brooke, R. Beckwith. 2004. Vineyard Computing: Sensor Networks in Agricultural Production. IEEE Jounal on Pervasive Computing 3(1): 38-45.

Caldeirinha, R. F. S. 2001. Radio Characterization of Single Tree at Micro- and Millimeter Wave Frequencies. PhD diss. South Wales: University of Glamorgan.

Campbell Scientific, I., ed. 2011. CR1000 Measurement and Control System Operator's Manual. 7/11 ed.

Costas, M. P., A. T. Theodore, P. Y. Constantine and C. K. Dimitris. 2012. Pest management control of olive fruit fly (Bactrocera oleae) based on a location-aware agro-environmental system. Computers and Electronics in Agriculture 8739-50.

Cumberland, B. C., J. S. Valacich and L. M. Jessup. 2004. Understanding Meteor Burst Communications Technologies. Communications of the ACM 47(1): 89-92.

Darr, M. J. and L. Zhao. 2008. A Model for Predicting Signal Transmission Performance of Wireless Sensor in Poultry Layer Facilities. Transactions of the ASABE 51(5): 1817-1827.

Desourdis, R. I., A. K. McDonough, S. C. Merrill, R. M. Bauman, D. A. Neumann, J. A. Lucas, D. Spector and D. E. Warren. 1994. Advanced Meteor-Burst Radio for Multimedia Communications. MILCOM 3685-689.

Edwards, R. and J. Durkin. 1969. Computer Prediction of Service Area for VHF Mobile Radio Networks. Proceedings of IEE. 166(9): 1493-1500.

European Cooperation in the Field of Scientific and Technical Research EURO-COST 231, Urban Transmission Loss Models for Mobile Radio in the 900 and 1800 MHz Bands. 1991.

Fukuda, A. K. Mukumoto, Y. Nagasawa et al. 2003. Experiments on Meteor Burst Communications in the Antarctic. Advances in Polar Upper Atmosphere Research 17:120-136.

Gupta S. 2012. Implantable Medical Devices – Cyber Risks and Mitigation Approaches. In Presentation at NIST Cyber Physical Systems Workshop.

Hagedorn, A., D. Starobinski and A. Trachtenberg. 2008. Rateless Deluge: Over-the-Air Programming of Wireless Sensor Networks Using Random Linear Codes. In International Conference on Information Processing in Sensor Networks, 457-466.

Hall, P.M., COST Project 235 Activities on Radiowave Propagation Effects on Next Generation Fixed Services Terrestrial Telecommunications Systems. 1993. Eighth International Conference on Antennas and Propagation, 2:655-659.

Ham, J.M., C. Williams, K. Shonkwiler. 2012. Estimating Ammonia Emissions from a Dairy Using Robotic Diffusive Samplers and Inverse Modeling. In Presentation at 2012 ASABE Annual International Meeting.

Han, W. 2011. Three-Tier Wireless Sensor Network Infrastructure for Environmental Monitoring. PhD diss. Manhattan, Kansas: Department of Biological and Agricultural Engineering.

Han, W., N. Zhang, Y. Zhang. 2008. A Two-Layer Wireless Sensor Network for Remote Sediment Monitoring. In Presentation at 2008 ASABE Annual International Meeting.

Hata, M. 1980. Empirical Formula for Propagation Loss in Land Mobile Radio Services. IEEE Transaction on Vehicular Technology 29(3): 317-325.

Healy, B. C., W. H. Shaw and J. R. Litko. 1989. A Modeling Perspective for Meteor Burst Communication. In 1989 Winter Simulation Conference, 1022-1031.

Hui, J. and D. Culler. 2004. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In SenSys '04 Proceedings of the 2nd international conference on Embedded networked sensor systems, 81-94. New York, NY.

Huma, Z., R. H. Nick, V. M. Geoff, R. Mark and C. Neil. 2013. The impact of agricultural activities on water quality: A case for collaborative catchment-scale management using integratedwireless sensor networks. Computers and Electronics in Agriculture 96126-138.

International Radio Consultative Committee. 1986. Recommendations and Reports of the CCIR, 1986: also Questions, Study Programmes, Resolutions, Opinions, and Decisions. International Telecommunication Union.

Johnson, D. E. 1987. Ten Years Experience with SNOTEL Meteor Burst Data Acquisition System. In Proc. Meteor Burst Communication Symposium, SII5-20.

Kim, K. D. Kang, S. W., Chung, S. O. et al. 2012. Remote Monitoring System for Greenhouse Environment using Mobile Devices. In Presentation at 2012 ASABE Annual International Meeting.

Koh, I. 2002. Advanced Diffraction and Wave Propagation Models for Characterization of Wireless Communication Channels. PhD diss. Ann Arbor, MI: University of Michigan.

Kulkarni, S. S. and L. Wang. 2005. MNP: Multihop Network Reprogramming Service for Sensor Networks. In 25th IEEE International Conference on Distributed Computing Systems, 7-16.

Lawlor, A., J. Torres, B. O'Flynn et al. 2012. DEPLOY: a Long Term Deployment of a Water Quality Sensor Monitoring System. Sensor Review 32(1): 29-38.

Leader, R. E. 1974. Meteor Burst Communication. In Presentation at Western Snow Conference.

Lee, Y. and C. Shen. 2007. A Transaction-based Approach to Over-the-air Programming in Wireless Sensor Networks. In International Symposium on Communications and Information Technologies, 1377-1382.

Liang, D., H. Gao, S. Li et al. 2012. A Real-Time Monitoring System for Cropland Based on Network Video. In Presentation at 2012 ASABE Annual International Meeting.

Li, A., M. Li, Z. Wang, M. Sun and H. Shen. 2012. A Wireless Sensor Network for Soil EC Measurement. In Presentation at 2012 ASABE Annual International Meeting.

Li, H., L. Zhao, M. J. Darr and P. Ling. 2009. Modeling Wireless Signal Transmission Performance Path Loss for ZigBee Communication Protocol in Residential Houses. In Presentation at 2009 ASABE Annual International Meeting.

Li, Z., N. Wang, A. Franzen et al. 2009. In-Field Soil Property Monitoring using Hybrid Sensor Network. In Presentation at 2009 ASABE Annual International Meeting.

Li, Z., T. Hong, N. Wang and T. Wen. 2010. Data Transmission Performance for 2.4GHz In-field Wireless Sensor Network. Computer Engineering and Technology 1465-469.

Mardeni, R. and L. Yih Pey. 2012. Path Loss Model Optimization for Urban Outdoor Coverage Using Code Diveision Multiple Access (CDMA) System at 822 MHz. Modern Applied Science 5(1): 20-32.

Mardeni, R. and T. Siva Priya. 2011. Optimized COST-231 Hata Models for WiMAX Path Loss Prediction in Suburban and Open Urban Environments. Modern Applied Science 4(9): 75-89.

Matschek, R. and B. Linot. 1999. Model for Wave Propagation in Presence of Vegetation Based on the UTD Associating Transmitted and Lateral Waves. National Conference on Antennas and Propagation.

Meng, Y. S., Y. H. Lee, B. C. Ng. 2009. Study of Propagation Loss Prediction in Forest Environment. Progress in Electromagnetics Research B 17: 117-133.

Meteor Burst Communication. 2006. Wikipedia. Available at: en.wikipedia.org/wiki/File: Meteor_Burst_SNOTEL.jpg.

Mickelson, K. D. 1989. Tracking 64,000 Vehicles with Meteor-Scatter Radios. Mobile Radio Technology24-38.

Oetting, J. D. 1980. An Analysis of Meteor Burst Communications for Military Applications. IEEE Transactions on Communications 28(9): 1591-1601.

Okumura, T., E. Hmori and K. Fukuda. 1968. Field Strength and Its Variability in VHF and UHF Land Mobile Service. Reivew Electrical Communication Laboratory 16(9-10): 825-873.

Paulsen, A. and A. Seville. 2000. Attenuation and Distortion of Millimeter Radio Wave Propagation through Vegetation. In Millennium Conference on Antennas and Propagation.

Pierce, F. J., T. V. Elliott. 2008. Regional and On-Farm Wireless Sensor Networks for Agricultural Systems in Eastern Washington. Computers and Electronics in Agriculture 61(1): 32-43.

Rappaport, T. S. and L. Milstein. 2002. Wireless Communications. Upper Saddle River: Prentice Hall.

Rice, P. L., A. G. Longley, K. A. Norton and A. P. Barsis. 1967. Transmission Loss Predictions for Tropospheric Communication Circuits. NBS Tech Note 2101.

Robert, W. C., J. D. Michael, B. Alan and H. Mark. 2013. Wireless sensor network with irrigation valve control. Computers and Electronics in Agriculture 9613-22.

Rogers, N. C., Seville, A., Richter, J., Ndzi, D., Savage, N., Caldeirinha, R. F. S., Shukla, A. K., Al-Nuaimi, M. O., Craig, K., Vilar, E., and Austin, J. 2002. A Generic Model of 1-60 GHz Radio Propagation through Vegetation. UK Radio-communications Agency AY3880/510005719.

Seville, A. 1997. Vegetation Attenuation: Modeling and Measurements at Millimetric Frequencies. In 10th International Conference on Antennas and Propagation, 14-17.

Stathopoulos, T., Heidemann, J. and Estrin, D. 2003. A Remote Code Update Mechanism for Wireless Sensor Networks. Center for Embedded Network Sensing.

Stephens, R. B. L., Al-Nuaimi, M. O. and Caldeirinha, R. F. S. 1998. Characterization of Depolarization of Radio Signals by Single Trees at 20 GHz. In Fifteenth National Radio Science Conference, 1-7. Helwan, Cairo, Egypt.

Stewart, R. L., J.F. Fox, C.K. Harnett. 2014. Estimating Suspended Sediment Concentration in Streams by Diffuse Light Attenuation. Journal of Hydraulic Engineering 140(8), 04014033.

Walfisch, J. and H. L. Bertoni. 1988. A Theoretical Model of UHF Propagation in Urban Environments. IEEE Transactions on Antennas and Propagation 36:1788-1796.

Wang, D., D. Agrawal, W. Toruksa, et al. 2010. Monitoring Ambient Air Quality with Carbon Monoxide Sensor-Based Wireless Network. Communications of the ACM 53(5): 138-141.

Wang, F. 2006. Physics-Based Modeling of Wave Propagation for Terrestrial and Space Communications. PhD diss. Ann Arbor, MI: University of Michigan.

Wang, N., N. Zhang and M. Wang. 2006. Wireless sensors in agriculture and food industry - Recent development and future perspective. Computers and Electronics in Agriculture 50(1): 1-14.

Wark, T., P. Corke, Sikka, P. Klingbei, et al. 2007. Transforming Agriculture through Pervasive Wireless Sensor Networks. IEEE Pervasive Computing 6(2): 50-57.

Weissberger, M. A. 1982. An Initial Critical Summary of Models for Predicting the Attenuation of radio waves by trees. ESD-TR-81-101.

Xiao, D., X. Jiang, J. Feng et al. A Wireless Multi-Sensor Network Deployment Framework for Wildland Fire Detection. In Presentation at 2012 ASABE Annual International Meeting.

Yick, J., B. Mukherjee, and D. Ghosal. 2008. Wireless Sensor Network Survey. Computer Networks 52(12): 2292-2330.

Zhang, H., N. Zhang, N. Wang et al. 2011. A General Agricultural Information Management Architecture for Distributed Wireless Sensor Network. In Presentation at 2011 ASABE Annual International Meeting.

Zhang, L. and D. Xiao. 2012. Collaborative image compression with error bounds in wireless sensor networks for crop monitoring. Computers and Electronics in Agriculture 891-9.

Zhou, Q., H. Zhang. 2013. A Self-Adaptive and Precise Oxygen Supplementary Equipment for Aquaculture with SMS Supported. In Presentation at 2013 ASABE Annual International Meeting.

Appendix A - Script file to configure the MBC radio

```
;SCAN Central Standard time CR1000 edited for KSU
;Engineering Unit Script
;PRIOR TO USING DO THE COMMAND:  FACTORY,DEFAULT,INIT
;545B SCAN
sched,del,all
freq,07
RXTH,-106
txlimit,200
checkin,1800
save
test
cls
assign,dta,1,cr1000,10
snp,ttl,0
snp,datap,C
RR,OFF
cr1000,maxq,20
cr1000,acqmode,all
cr1000,interval,30
cr1000,time,cr1000
cr1000,group,cr1000
CR1000,SCALE,cr1000
cr1000,modem enable,on
destination,0
;Set timezone to central standard time
timezone,0,2
sched,t,4:50:15,updt,time
pakbus,id,2
sched,i,1:0:0,start
id,6052,0002,multi,init
save
```

Appendix B - CR1000 datalogger program running at MBC central station

```
' CR1000 Series Datalogger
' program author: Xu Wang
' RF401 is attached to CR1000 via RS232 port, while MCC545B is attached to CR1000 via CSI/O port.
' CR1000 receives a data array of 40 float type variables every minute via RS232 port.
' Then CR1000 outputs 4 strings with 10 float variables in each of them via CSI/O port to MCC545B.
'Declare Public Variables
Public mDest1(10),mDest2(10),mDest3(10),mDest4(10)
Public Batt_volt
Public snrid
Public seqnum
Public sec1
Public sec2
Public crc1
Public crc2
'Declare Tables
DataTable(Tb1,true,-1)
  DataInterval(0,5,Min,10)
  seqnum = mDest1(2)
  mDest1(2) = INT(seqnum) INTDV 7999
  sec1 = mDest1(4)
  mDest1(4) = INT(sec1) INTDV 3600
    Sample(1,snrid,FP2)
  Sample(1,mDest1(1),FP2)
  Sample(1,mDest1(2),FP2)
  Sample(1,mDest1(3),FP2)
  Sample(1,mDest1(4),FP2)
  Sample(1,mDest1(5),FP2)
  Sample(1,mDest1(6),FP2)
  Sample(1,mDest1(7),FP2)
  Sample(1,mDest1(8),FP2)
  Sample(1,mDest1(9),FP2)
  Sample(1,mDest1(10),FP2)
EndTable

DataTable(Tb2,true,-1)
  DataInterval(0,5,Min,10)
  crc1 = mDest2(10)
  mDest2(10) = INT(crc1) INTDV 7999
  mDest2(9) = INT(crc1) MOD 7999
  mDest2(8) = INT(sec1) MOD 3600
```

```
  mDest2(7) = INT(seqnum) MOD 7999
  Sample(1,mDest2(1),FP2)
  Sample(1,mDest2(2),FP2)
  Sample(1,mDest2(3),FP2)
  Sample(1,mDest2(4),FP2)
  Sample(1,mDest2(5),FP2)
  Sample(1,mDest2(6),FP2)
  Sample(1,mDest2(7),FP2)
  Sample(1,mDest2(8),FP2)
  Sample(1,mDest2(9),FP2)
  Sample(1,mDest2(10),FP2)
EndTable

DataTable(Tb3,true,-1)
  DataInterval(0,5,Min,10)
  sec2 = mDest3(2)
  mDest3(2) = INT(sec2) INTDV 3600
  Sample(1,mDest3(1),FP2)
  Sample(1,mDest3(2),FP2)
  Sample(1,mDest3(3),FP2)
  Sample(1,mDest3(4),FP2)
  Sample(1,mDest3(5),FP2)
  Sample(1,mDest3(6),FP2)
  Sample(1,mDest3(7),FP2)
  Sample(1,mDest3(8),FP2)
  Sample(1,mDest3(9),FP2)
  Sample(1,mDest3(10),FP2)
EndTable

DataTable(Tb4,true,-1)
  DataInterval(0,5,Min,10)
  mDest4(5) = INT(sec2) MOD 3600
  crc2 = mDest4(8)
  mDest4(7) = INT(crc2) MOD 7999
  mDest4(8) = INT(crc2) INTDV 7999
  Sample(1,mDest4(1),FP2)
  Sample(1,mDest4(2),FP2)
  Sample(1,mDest4(3),FP2)
  Sample(1,mDest4(4),FP2)
  Sample(1,mDest4(5),FP2)
  Sample(1,mDest4(6),FP2)
  Sample(1,mDest4(7),FP2)
  Sample(1,mDest4(8),FP2)
  Sample(1,mDest4(9),FP2)
  Sample(1,Batt_volt,FP2)
EndTable
```

```
'Main Program
BeginProg
  'Open RS232 port with 100 bytes in buffer
  SerialOpen(ComRS232,9600,0,0,100)
  snrid=2914
        Scan (30,Sec,3,0)
          'Send beacon signal to another datalogger, ready to receive data
          'RF401 is attached to CR1000 via RS232 port
          'CSI/O (as ComSDC7 or ComSDC8) is reserved for MBC545B radio
    'SendVariables(Res,ComRS232,6,6,0,300,"Public","f6",flag_cr1000,1)
    Battery (Batt_volt)
    CallTable Tb1
    CallTable Tb2
    CallTable Tb3
    CallTable Tb4
        NextScan
EndProg
```

Appendix C - Java server programs

```
// KSUServer.java

import java.net.*;
import java.io.*;
public class KSUServer {
	/**
	 * This program runs in the server computer to receive
	 * data from cellular modem, and to send the program
	 * file to the cellular modem.
	 */
	public static void main(String[] args) throws IOException {
		ServerSocket serverSocket = null;
		boolean listening = true;
		try {
			serverSocket = new ServerSocket(2013);
		} catch (IOException e) {
			System.err.println("Could not listen on port: 2013.");
			System.exit(-1);
		}
		System.out.println("Waiting for a connection...");
		while (listening) {
			new KSUMultiServerThread(serverSocket.accept()).start();
		}
		serverSocket.close();
	}
}
```

```
// KSUMultiServerThread.java

import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import java.text.*;

public class KSUMultiServerThread extends Thread {
	private Socket socket = null;
	public KSUMultiServerThread(Socket socket) {
		super("KSUMultiServerThread");
		this.socket = socket;
		System.out.println("Accepted a connection from: "+ socket.getInetAddress());
	}
	public void run() {
		try {
			PrintWriter out = new PrintWriter(System.out, true);
			BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
			// For output to datalogger
			BufferedWriter bwout = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
			String resp = "CONN\r\n";
			String srcFilePath = "c:/toSend";
			File srcPath = new File(srcFilePath);
			File srcFile = null;
			Boolean fexist = false;
			String srcFullPath = "";
			String srcFileName = "";
			final int Num_Records = 2;
			int i=0,k=0,j=0;
			int ptn=0,pto=0;
			boolean[] v = new boolean[Num_Records];
			double vdist = 0.04;
			int vsamp = 280;
			int[] intdata = new int[40];
			int[] crcRemote = new int[Num_Records];
			String[] results = new String[Num_Records];
			String[] crcCheck = new String[Num_Records];
			String[] strCheck = new String[Num_Records];
			String strSQL = "";
			String inputLine, token;
			Calendar c = Calendar.getInstance();
			int year = 0,days=0;
			NumberFormat formatsensorbatt = new DecimalFormat("0.00");
			NumberFormat formatsensordata = new DecimalFormat("0.0000");
```

```
                        NumberFormat formatradiobatt = new DecimalFormat("00.00");
                        NumberFormat formatvelocity = new DecimalFormat("000.00");
                        CRC16 my_crc = new CRC16();
                        results[0]=results[1]="";
                        crcCheck[0]=crcCheck[1]="";
                        crcRemote[0]=crcRemote[1]=0;
                        v[0]=v[1]=false;
                        while ((inputLine = in.readLine()) != null) {
                                inputLine =
inputLine.substring(inputLine.indexOf("$")+1,inputLine.indexOf("#"));
                                out.println(inputLine);
                                Connection conn = null;
                                // For OTA
                                if (srcPath.exists() && srcPath.isDirectory() && !fexist){
                                        if (srcPath.listFiles().length > 0){
                                                String[] srcFlst = srcPath.list();
                                                for (int fi=0;fi<srcFlst.length;fi++){
                                                        srcFullPath = srcFilePath + "/" + srcFlst[fi];
                                                        out.println(srcFullPath);
                                                        srcFileName = srcFlst[fi];
                                                        srcFile = new File(srcFullPath);
                                                }
                                                fexist = true;
                                                out.println("File found.");
                                        }
                                }
                                if (fexist) {
                                        if (inputLine.equalsIgnoreCase("Ready")){
                                                out.println("Ready to send "+ srcFullPath + "...");
                                                DataInputStream fis = new DataInputStream(new
BufferedInputStream(new FileInputStream(srcFullPath)));
                                                DataOutputStream fout = new
DataOutputStream(socket.getOutputStream());
                                                String tempstr;
                                                int bufferSize = 1024;
                                                byte[] buf = new byte[bufferSize];
                                                fout.writeBytes("OK$$");
                                                while (true){
                                                        int read = 0;
                                                        if (fis != null){
                                                                read = fis.read(buf);
                                                        }
                                                        if (read == -1) {
                                                                break;
                                                        }
                                                        for (int si=0;si<read;si++) {
```

```
                                                    if ((buf[si]+256)%256 < 16) {
                                                        tempstr =
"0"+(Integer.toHexString((buf[si]+256)%256)).toUpperCase();
                                                    }
                                                    else {
                                                        tempstr =
(Integer.toHexString((buf[si]+256)%256)).toUpperCase();
                                                    }
                                                    out.print(tempstr+" ");
                                                    fout.writeBytes(tempstr+" ");
                                                }
                                            }
                                            fout.writeBytes("$$#");
                                            fout.flush();
                                            fis.close();
                                            fexist = false;
                                            srcFile.delete();
                                        }
                                        else {
                                            bwout.write(resp + srcFileName + "\r\n" );
                                            bwout.flush();
                                        }
                                    }
                                }
                                else {
                                    if (!inputLine.equalsIgnoreCase("O")){
                                        inputLine = inputLine + " #";
                                        StringTokenizer st = new StringTokenizer(inputLine);
                                        search:
                                            while (st.hasMoreTokens()) {
                                                token=st.nextToken();
                                                //Check the beginning
                                                if (token.equals("$")){
                                                    results[k] = "";
                                                    crcCheck[0] = "";
                                                    crcRemote[0] = 0;
                                                    i = 0;
                                                }
                                                if (i==39 || i==40){
                                                    intdata[i-1]  =
(int)(Float.valueOf(token).floatValue()*100.0);
                                                    if (intdata[i-1]>5000) intdata[i-1] =
5000;
                                                }
                                                //Validate data and convert String to integer
                                                if (i > 0 && i < 41){
```

```
                                                        if (!(i==39 || i==40)){
                                                                try {
                                                                        intdata[i-1] =
Integer.parseInt(token);
                                                                }
                                                                catch
(NumberFormatException nfe) {
                                                                        out.println("Invalid
data");
                                                                        break search;
                                                                }
                                                        }
                                                        //prepare string for crc check
                                                        if (!(i < 3 || i ==20 || i > 37)){//do not
check gateway, repeater and central battery as well as crc data
                                                                crcCheck[k] = crcCheck[k] +
token;
                                                        }
                                                        switch (i) {
                                                        case 39: //Gateway battery
                                                        case 40: //Repeater battery
                                                                break;
                                                        case 3:
                                                                if (intdata[i-1]<0 || intdata[i-
1]>366){

        results[0]=results[1]="";

                                                                        if (conn != null){
                                                                                try{
                                                                                conn.close ();
                                                                                }catch
(Exception e) { // ignore close errors
                                                                                }
                                                                        }
                                                                        break search;
                                                                }
                                                                year =
c.get(Calendar.YEAR); //get year
                                                                days = intdata[i-1];
                                                                JulianDate JD = new
JulianDate(year, days);
                                                                JD.convert();
                                                                results[k] = results[k] + "" +
JD.cDate + ", ";
                                                                break;
                                                        case 21:
```

```
                                                    if (intdata[i-1]<0 || intdata[i-
1]>366){

        results[0]=results[1]="";
                                                            if (conn != null){
                                                                try{


        conn.close ();
        }catch (Exception e) { // ignore close errors
                                                                }
                                                            }
                                                            break search;
                                                    }
                                                    year =
c.get(Calendar.YEAR); //get year

                                                    days = intdata[i-1];
                                                    JulianDate JD2 = new
JulianDate(year, days);

                                                    JD2.convert();
                                                    results[k] = results[k] +
intdata[0] + ", " + intdata[1] + ", '"+ JD2.cDate + ", ";
                                                    break;
                                                case 4:
                                                case 22:
                                                    if (intdata[i-1]<0 || intdata[i-
1]>86400){

        results[0]=results[1]="";
                                                            if (conn != null){
                                                                try
                                                                {


        conn.close ();

        //System.out.println ("Database connection terminated");
                                                                }catch
(Exception e) { // ignore close errors
                                                                }
                                                            }
                                                            break search;
                                                    }
                                                    TimeString TS = new
TimeString(intdata[i-1]);

                                                    TS.convert();
                                                    results[k] = results[k] +
TS.ctime + "', ";
```

```
                                                        break;
                                                case 5://check group ID
                                                case 23:
                                                        if (intdata[i-1]==125 ||
intdata[i-1]==22){

                                                                v[k] = false;
                                                        }else{
                                                                v[k] = true;
                                                        }
                                                        results[k] = results[k] +
intdata[i-1] + ", ";

                                                        break;

                                                case 6: // Get mote ID
                                                case 24:
                                                        results[k] = results[k] +
intdata[i-1] + ", ";

                                                        break;
                                                case 7: //Mote battery
                                                case 25:
                                                        if (v[k]){
                                                                results[k] = results[k]
+ formatsensordata.format(intdata[i-1]*2.5/4096.0) + ", ";
                                                        }else{
                                                                results[k] = results[k]
+ formatsensorbatt.format(intdata[i-1]/1000.0) + ", ";
                                                        }
                                                        break;
                                                case 15: //Rain gauge
                                                case 33:
                                                        if(v[k]){
                                                                results[k] = results[k]
+ formatsensordata.format(intdata[i-1]*2.5/4096.0) + ", ";
                                                        }else{
                                                                if (intdata[i-1] != 0){
                                                                        ptn =
intdata[i-1];
                                                                        if ((ptn-pto) >
10){

        results[k] = results[k] + formatsensordata.format(0) + ", ";
                                                                                pto =
ptn;
                                                                        }else {

        results[k] = results[k] + formatsensordata.format(ptn-pto) + ", ";
```

```
                                                                pto =
ptn;
                                                                    }
                                                                }else results[k] =
results[k] + formatsensordata.format(0) + ", ";
                                                        }
                                                        break;
                                                case 16:// Thermocouple
                                                case 34:
                                                        if(v[k]){
                                                                results[k] = results[k]
+ formatsensordata.format(intdata[i-1]*2.5/4096.0) + ", ";
                                                        }else{
                                                                results[k] = results[k]
+ formatradiobatt.format((intdata[i-1]*2.5/4096.0/0.2473-0.011)*25) + ", ";
                                                        }
                                                        break;
                                                case 17: //Velocity
                                                case 35:
                                                        if(v[k]){
                                                                results[k] = results[k]
+ formatsensordata.format(intdata[i-1]*2.5/4096.0) + ", ";
                                                        }else{
                                                                if (intdata[i-1] > 1){
                                                                        results[k] =
results[k] + formatvelocity.format(vdist*vsamp/intdata[i-1]) +", ";
                                                                }else{
                                                                        results[k] =
results[k] + formatvelocity.format(0.0) +", ";
                                                                }
                                                        }
                                                        break;
                                                case 8:
                                                case 9:
                                                case 10:
                                                case 11:
                                                case 12:
                                                case 13:
                                                case 14:
                                                case 18:
                                                case 19:
                                                case 26:
                                                case 27:
                                                case 28:
                                                case 29:
                                                case 30:
```

```
                                        case 31:
                                        case 32:
                                        case 36:
                                        case 37:
                                                results[k] = results[k] +
formatsensordata.format(intdata[i-1]*2.5/4096.0) + ", ";
                                                break;
                                        case 20:
                                                crcRemote[k] = intdata[i-1];
//CRC LWSN 1
                                                k = 1;
                                                results[k]="";
                                                crcCheck[k]="";
                                                crcRemote[k]=0;
                                                break;
                                        case 38:
                                                crcRemote[k] = intdata[i-1];
//CRC LWSN 2
                                                k = 0;
                                                break;
                                        default:
                                                results[k] = results[k] +
intdata[i-1] + ", ";
                                                break;
                                        }
                                }
                                //Check the end
                                if (token.equals("#")){
                                        for (j=0; j<Num_Records; j++){
                                                out.println(results[j]);
                                                if (crcRemote[j] ==
my_crc.getValue(crcCheck[j],crcCheck[j].length())){
                                                        out.println("Ok");
                                                        strCheck[j] = "0";
                                                }else{
                                                        out.println
("Checksum is " + Long.toString(my_crc.getValue(crcCheck[j],crcCheck[j].length())));
                                                        strCheck[j] = "1";
                                                }
                                        }
                                        try
                                        {
                                                String userName = "root";
                                                String password = "bae840-
05";
```

```
                                                    String url =
"jdbc:mysql://129.130.45.77:3308/estcp";
                                                    Class.forName
("com.mysql.jdbc.Driver").newInstance ();
                                                    conn =
DriverManager.getConnection (url, userName, password);
                                                    //System.out.println
("Database connection established");
                                                    Statement s =
conn.createStatement ();
                                                    for (j=0; j<Num_Records;
j++){
                                                            if (v[j]){
        strSQL="INSERT INTO tblvelocity_riley (PakBusAdd, SeqNum, DateTime, PacketID,
MoteID,"
        + " Ch0, Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Ch7, Ch8, Ch9,"
        + " Ch10, Ch11, Ch12, CRC)"
        + " VALUES(" + results[j] + strCheck[j] + ")";
                                                            }else{
        strSQL="INSERT INTO tblresults (PakBusAdd, SeqNum, DateTime, GroupID,
MoteID,"
        + " MoteBatt, Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Ch7, Ch8, Ch9,"
        + " Ch10, Ch11, Ch12, Ch14, GatewayBatt, RepeaterBatt, CenterBatt)"
        + " VALUES("
        + results[j]
                + strCheck[j] + ", "
                + formatradiobatt.format(intdata[38]/100.0) + ", "
                + formatradiobatt.format(intdata[39]/100.0) + ", "
                + formatradiobatt.format(0) + ")";
        s.executeUpdate (strSQL);
                                                            }
                                                    }
```

```
                                                    s.close ();
                                                }
                                                catch (Exception e){
                                                    System.err.println(e);
                                                    System.err.println ("Cannot
connect to database server");
                                                }
                                                finally
                                                {
                                                    if (conn != null)
                                                    {
                                                        try
                                                        {
                                                            conn.close ();

        //System.out.println ("Database connection terminated");
                                                        }
                                                        catch (Exception e)
{ // ignore close errors
                                                        }
                                                    }
                                                }
                                                out.println("Data seq # " + intdata[1]
+ " received successfully.\n");
                                                i=0;
                                            }
                                            i++;
                                        }
                                    }
                                }
                            }
                            out.close();
                            in.close();
                            bwout.close();
                            socket.close();
        } catch (IOException e) {
                if (e instanceof SocketException){
                    //System.out.println("Connection reset.");
                    System.out.println("Waiting for a connection...");
                }else{
                    e.printStackTrace();
                }
        }
}
}
```

Appendix D - CR1000 Datalogger program at central station

```
'CR1000 Series Datalogger
'program author: Xu Wang
'Pakbus Address 2

'Variables for basic program
Public Res
Public Dest1(10),Dest2(10),Dest3(10),Dest4(10)
Public flag
Public rawData As String * 1024
Public i,j
'Variables for OTA Rx from cellular modem
Public OpenFile1 As Long, StringVal(2) As String * 30720
Public StringValt As String *30720
Public SigVal As String * 500
Public Filename As String *50
Public CloseStat
Public ResultString(5) As String * 20
Public StrComm As String
'Variables for OTA Tx to repeater station
Public StringSrc As String * 10240
Public ResultStr(10) As String
Public TxStr(10)
Public ReadFile1,SPoint
Public CTTxDone
Public CTTxReady
Public RXResponse
Public TxC
Public flag7_1

Sub GetData
  rawData = "$$ "
          For i=1 To 10
    rawData = rawData + Dest1(i) + " "
  Next
  If Dest2(10)=5120 Then
    Delay(1,1,sec)
  EndIf
  For i=1 To 10
    rawData = rawData + Dest2(i) + " "
  Next
  If Dest3(10)=5120 Then
    Delay(1,1,sec)
  EndIf
  For i=1 To 10
```

```
    rawData = rawData + Dest3(i) + " "
  Next
  If Dest4(10)=5120 Then
    Delay(1,1,sec)
  EndIf
  For i=1 To 10
    rawData = rawData + Dest4(i) + " "
  Next
  rawData = rawData + "##"
  SerialOpen(ComRS232,9600,0,0,500)
          SerialFlush(ComRS232)
  SerialOut(ComRS232,rawData+CHR(13),"",0,100)
  rawData=""
  For i=1 To 9
    Dest1(i) = 0
    Dest2(i) = 0
    Dest3(i) = 0
    Dest4(i) = 0
  Next
  Dest1(10)=5120
  Dest2(10)=5120
  Dest3(10)=5120
  Dest4(10)=5120
  i=0
EndSub

'Main Program
BeginProg
  flag=1
  Dest1(10)=5120
  Dest2(10)=5120
  Dest3(10)=5120
  Dest4(10)=5120
  rawData=""
  flag7_1 = 0
  SPoint = 0
  CTTxDone = 1
  CTTxReady = 1
  RXResponse = -1
  TxC = 1000
  SetStatus("USRDriveSize",153600)
  SerialOpen(ComRS232,9600,0,0,500)
  Delay(1,3,Sec)
  SerialOpen(ComSDC8,9600,0,0,30270)
  SigVal=""
  StrComm="$O#"
```

```
      Scan (1,Sec,0,0)
' Sediment data processing
' State 0
      SendVariables(Res,ComSDC8,0,6,00000,100,"Public","f6_1",flag,1)
      Delay(1,1,sec)
      If Dest1(10)<>5120 Then
        Call GetData
EndIf
' Ready to receive file from server
' State 1
      SerialFlush(ComRS232)
      SerialOut(ComRS232, StrComm+CHR(13),"",0,50)   ' Output "O" to server
SerialIn(SigVal,ComRS232,200,0,50)            ' Read the output from server
' Process SigVal, look for signal of "Conn" and file name
SplitStr(ResultString(),SigVal,CHR(13)+CHR(10),5,5)
SigVal=""                                     ' Clean SigVal
If ResultString(1)="CONN" AND TxC=1000 Then
  ' State 2
  StrComm = "$Ready#"
  SerialFlush(ComRS232)
  ' Output "Ready" to server
  SerialOut(ComRS232,StrComm+CHR(13),"",0,100)
  SerialFlush(ComRS232)
  StringValt = ""
  SerialIn(StringValt,ComRS232,1000,CHR(35),30720)
  ' Seperate StringVal, look for "OK"
  SplitStr(StringVal,StringValt,"$$",2,5)
  ' Save file in memory
  If StringVal(1)="OK" Then
    Filename = "USR:"+ResultString(2)
    OpenFile1 = FileOpen(Filename,"a",-1)
    FileWrite(OpenFile1,StringVal(2)+" # ",0)
    CloseStat = FileClose(OpenFile1)
  EndIf
  ' If Stargate program, further send to Repeater, or reload
  If ResultString(2)="prelogger" Then
    TxC = 0
  Else
    FileManage(Filename,4)
  EndIf
  StringValt = ""
  StringVal(1) = ""
  StringVal(2) = ""
  SigVal = ""
  SerialFlush(ComRS232)
  StrComm = "$O#"
```

```
    ResultString(1)=""
    ResultString(2)=""
   ElseIf TxC<1 Then
    ' State 3 Send file to repeater station
    SendVariables(RXResponse,ComSDC8,0,6,00000,50,"Public","RPRxReady",CTTxReady,1)
    Delay(1,1,sec)
                If flag7_1=1 Then
                  If CTTxDone<>0 Then
                    OpenFile1 = FileOpen(Filename,"r",SPoint)
                    ReadFile1 = FileRead(OpenFile1,StringSrc,30)
                    CloseStat = FileClose(OpenFile1)
                    SplitStr(ResultStr(),StringSrc,CHR(32),10,5)
                    For i = 1 To Ceiling(ReadFile1/3)
                                If StrComp(ResultStr(i),"#")<>0 Then
                                  TxStr(i) = HexToDec(ResultStr(i))
                                  Else
                                  TxStr(i) = 1000
                                EndIf
                        Next
        CTTxDone=0
                    EndIf
     SendVariables(RXResponse,ComSDC8,0,6,00000,100,"Public","file()",TxStr(),10)
     Delay(1,2,sec)
     If CTTxDone<>0 Then
      SPoint = SPoint+ReadFile1
      StringSrc=CHR(0)
      For i = 1 To 10
       TxStr(i) = 0
      Next
     EndIf
     flag7_1=0
     If ReadFile1<30 Then
      SPoint = 0
      TxC = 1000
      FileManage(Filename,8)
      StrComm = "$O#"
     EndIf
    EndIf
   EndIf
        NextScan
EndProg
```

Appendix E - CR206 Datalogger program at repeater station

```
'CR200 Series Datalogger
'program author: Xu Wang
'PakBus Address 6

'Variables for basic program
Public Res
Public fDest1(10),fDest2(10),fDest3(10),fDest4(10)
Public file(10)
Public flag, f6_1
Public batt_volt
'Variables for OTA
Public RPRxReady
Public RPRxDone
Public RPTxStart
Public RPTxDone
Public i,s
'Define Subroutines
Sub Send_Data
	Res=-1
		While Res<>0
		  SetValue(Res,fDest1,10,Dest1,1,2,2,00000)
		Wend
		Res=-1
		If fDest2(10)=5120
		  Delay(3,sec)
		EndIf
		While Res<>0
		  SetValue(Res,fDest2,10,Dest2,1,2,2,00000)
		Wend
		Res=-1
		If fDest3(10)=5120
		  Delay(3,sec)
		EndIf
		While Res<>0
		  SetValue(Res,fDest3,10,Dest3,1,2,2,00000)
		Wend
		Res=-1
		If fDest4(10)=5120
		  Delay(3,sec)
		EndIf
		'voltage for repeater
	fDest4(10)=batt_volt
	While Res<>0
		SetValue(Res,fDest4,10,Dest4,1,2,2,00000)
```

```
        Wend
        Res=-1
      flag=1
                fDest1(1)=0
                fDest2(10)=5120
                fDest3(10)=5120
                fDest4(10)=5120
EndSub

'Main Program
BeginProg
  flag=1
  Res=-1
  RPRxDone=1
  RPRxReady=0
  RPTxStart=0
  RPTxDone=0
  i=0
  s=1
        Scan (1,Sec)
   f6_1=0
   ' Sediment data Tx/Rx
   SetValue(Res,flag,1,f1_1,1,1,1,00000)
   Delay(1,sec)
   If fDest1(1)<>0 Then
     flag=0
     While f6_1=0
     Wend
     Battery (batt_volt)
     Call Send_Data
     SetValue(Res,flag,1,f1_2,1,1,1,00000)
   EndIf
   ' Receive file data from central station
   If RPRxReady<>0 AND s=1 Then
                SetValue(Res,flag,1,flag7_1,1,2,2,00000)
          Delay(2,sec)
          ' Check non-empty
          For i = 1 To 10
                If file(i)<>0 Then
                  SetValue(Res,RPRxDone,1,CTTxDone,1,2,2,00000)
                  s=2
                  ExitFor
                EndIf
              Next
              RPRxReady = 0
        EndIf
```

```
        ' Send file data to gateway station
   If s=2 Then
                SetValue(Res,flag,1,GWRxStart,1,1,1,00000)
                Delay(1,sec)
                If RPTxStart<>0 Then
     SetValue(Res,file(1),10,gfile,1,1,1,00000)
     RPTxStart=0
     Delay(2,sec)
     If RPTxDone<>0 Then
      RPTxDone=0
      s=1
      For i = 1 To 10
                  file(i)=0
                Next
     EndIf
    EndIf
   EndIf
        NextScan
EndProg
```

Appendix F - CR206 Datalogger program at gateway station

```
' CR200 Series Datalogger
' program author: Xu Wang
' PakBus Address 1

' Variables for basic program
Public SI_in(40)
Public Response
Public batt_volt
Public f1_1,f1_2
' Variables for OTA
Public gfile(10)
Public ogfile(5)
Public GWRxStart
Public GWRxDone
Public ref
Public i
' Declare Constants
Const MAXVALUES = 40
Const TERMCHAR = 35

'Main Program
BeginProg
  Delay(60, sec)
          Delay(60, sec)
          Delay(60, sec)
  RealTime( SI_in , 0 )
  Print(-2,9600,SI_in(1),"/",SI_in(2),"/",SI_in(3))
  Print(-2,9600,"/", SI_in(4),"/", SI_in(5), "/")
  Print(-2,9600, SI_in(6),CHR$(13))
          SI_in(1)=0
          Response=-1
  GWRxDone=1
  GWRxStart=0
  ref=0
  f1_1=0
  f1_2=1

Scan (1,Sec)
            ' Sediment data processing
              f1_1=0
     Print(2,9600)
     SerialInput(SI_in(),MAXVALUES,TERMCHAR,$)
     If SI_in(1)<>0 Then
       If SI_in(1)=4000 Then 'time synchronization
```

```
      RealTime( SI_in , 0 )
      Print(-2,9600,SI_in(1),"/",SI_in(2),"/",SI_in(3))
      Print(-2,9600,"/", SI_in(4),"/", SI_in(5), "/")
      Print(-2,9600, SI_in(6),CHR$(13))
      SI_in(1)=0
    Else
      While f1_1=0
      Wend
      Battery (batt_volt)
      SI_in(39) = batt_volt
      Response=-1
      While Response<>0
       SetValue(Response,SI_in(1), 10,fDest1,1,6,6,00000)
      Wend
      Response=-1
      While Response<>0
       SetValue(Response,SI_in(11),10,fDest2,1,6,6,00000)
      Wend
      Response=-1
      While Response<>0
       SetValue(Response,SI_in(21),10,fDest3,1,6,6,00000)
      Wend
      Response=-1
      While Response<>0
       SetValue(Response,SI_in(31),10,fDest4,1,6,6,00000)
      Wend
      SI_in(1)=0
      Print(-2,9600,f1_2,CHR$(13))
    EndIf
  EndIf
' Receive data from repeater station
f1_1=1
      If GWRxStart<>0 Then
       SetValue(Response,f1_1,1,RPTxStart,1,6,6,00000)
       ' Check data non-empty
              For i=1 To 10
               If gfile(i)<>0 Then
                ref=1
                ExitFor
               EndIf
              Next
              If ref=1 Then
               ' Check data not the same as last time
               For i=1 To 5
                If ogfile(i)<>gfile(i) Then
                 Response=-1
```

```
                    SetValue(Response,GWRxDone,1,RPTxDone,1,6,6,00000)
                    ref=2
                    ExitFor
                  EndIf
                      Next
                      For i=1 To 5
                        ogfile(i)=gfile(i)
                      Next
                      If ref=2 Then
                  ' Send signal to Stargate
                  Print(-2,9600,f1_2,CHR$(13))
                  ' Receive signal from Stargate
                  SerialInput(SI_in(),MAXVALUES,TERMCHAR,$)
                  If SI_in(1)=2000 Then
                  Print(2,9600,gfile(1),gfile(2),gfile(3),gfile(4),gfile(5))
                  Print(2,9600,gfile(6),gfile(7),gfile(8),gfile(9),gfile(10))
                 EndIf
                 GWRxStart=0
                 For i = 1 To 10
                  gfile(i)=0
                 Next
                 ref=0
                EndIf
    EndIf
   EndIf
        NextScan
EndProg
```

Appendix G - Stargate program

```
/* This program is used to prepare data before transmit them
to CR206 via RS-232 cable

Version 2.0 demo by : Xu Wang
Date: December 14, 2010 16:30:00 PM
1. remove velocity data processing
2. remove the time stamp checking, keep time counter,
   still update datetime
3. use semaphore to judge the new incoming data
4. 1 records per packet

Version 2.1 demo by : Xu Wang
Date: December 14, 2010 16:30:00 PM
1. add velocity data processing
2. 8s per data
3. no response

Version 2.4 by : Xu Wang
Date January 3, 2011 10:30:00 AM
1. Queue length to 124

Version 3.0 by : Xu Wang
Date January 14, 2011 11:20:00 AM
1. Keep velocity data processing and sending velocity raw data back
2. Use semaphore not time stamp to distinguish new incoming data
3. Still keep updating datetime
4. 2 records per packet
5. Use Response and timeout 2 methods to decide whether to deliver the new data
6. Add Storing Queue and Ready Queue
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include "crc.h"

#define LOGGER "/mnt/cf1/mydata/data_for_logger"
// Semahpore
#define SEMAPHORE "/mnt/cf1/mydata/semaphore"

#define MAX_CH 13
#define S_QUEUE_LEN 450
```

```
int JulianDate (int,int,int);
char *trim_nocopy (char *);
int main(int argc, const char *argv[])
{
        FILE *f1, *f2, *f3;
        char buffer[256];
        char logger[256];
        char semaphore[256];
        char smph[2];
        char str[4];
        char strDate[20];
        char *pch;
        char cdate[6][10];
        char strGroupID[3],strMoteID[3],strPacketID[3],strReserved[3];
        char strCh[5];
        int intCh[MAX_CH];
        int year, month, day, hour, minute, second;
        int i,j;
        int PakBusAdd = 1;
        int seqnum = 1;
        int intGroupID, intMoteID, intPacketID, intReserved;
        int timecount = 0;
        long time;
        int l,n,m=0,M;
        float record[S_QUEUE_LEN][18];
        int head=0,tail=0;
        float readyqueue[40];
        char *tmp = " ";
        unsigned char crctemp[10];
        unsigned char crcString[300];
        int result;
        int resdg=0;
        int rep;
        int tout = 30;
        int tg[10];
        int txdone=1;
        char c;

        if (argc == 2) {
                strcpy(logger, argv[1]);
                strcpy(semaphore, argv[1]);
                strcat(logger, LOGGER); // " argv[1]/mnt/cf1/mydata/data_for_logger"
                strcat(semaphore, SEMAPHORE); // "argv[1]/mnt/cf1/mydata/semaphore"
        } else {
                strcpy(logger, LOGGER);
```

```
        strcpy(semaphore, SEMAPHORE);
    }

    crcInit();//initial for crc check

    for(;;){
        resdg = getResponse(2);
        if (resdg == 2) {
            if (!(f3 = fopen("../home/xbow/apps/prelogger/prelogger1","w"))){
                perror("File cannot be opened. \n");
                exit(1);
            }
            while(txdone){
                // printf("Input numbers: \n");

    scanf("%d %d %d %d %d %d %d %d %d %d%c",&tg[0],&tg[1],&tg[2],&tg[3],&tg[4],
&tg[5],&tg[6],&tg[7],&tg[8],&tg[9],&c);
                for (i=0;i<10;i++){
                    if (tg[i]>255) {
                        txdone=0;
                        fflush(f3);
                        fclose(f3);
                        break;
                    }
                    else {
                        fwrite(&tg[i],1,1,f3);
                    }
                }
                fflush(stdin);
            }
            system("../home/xbow/apps/prelogger/restpre.sh");
            exit(0);
        }else {
            // Check semaphore
            if (!(f2 = fopen(semaphore, "r+"))) {
                perror ("Couldn't read semaphore value. \n");
                exit(1);
            }
            fgets(buffer,sizeof(buffer),f2);
            fflush(f2);
            fclose(f2);

            memset(smph,'\0',2);
            strncpy(smph,buffer,1);
            //printf("Pre-semaphore is %s\n",smph); //test
            if (strcmp(smph,"1") == 0){
```

```
// There is new data, start to process
//printf("Start to process data...\n"); //test
if (!(f1 = fopen(logger, "r+"))) {
        perror ("Couldn't open data_for_logger file.\n");
        exit(1);
}
while(fgets(buffer,sizeof(buffer),f1) != NULL ){
        memset (str, '\0',4);
        strncpy (str,buffer,3);
        if(strcmp(str,"[20") == 0){
                memcpy(strDate,buffer+1,19);
                pch = strtok (strDate,"/ :");
                i=0;
                while (pch !=NULL){
                        strcpy(cdate[i],pch);
                        i++;
                        pch = strtok (NULL, "/ :");
                }
                year = strtol(cdate[0],NULL,10);
                month = strtol(cdate[1],NULL,10);
                day = strtol(cdate[2],NULL,10);
                day = JulianDate (year, month, day);
                hour = strtol(cdate[3],NULL,10);
                minute = strtol(cdate[4],NULL,10);
                second = strtol(cdate[5],NULL,10);
                time=hour*3600+minute*60+second;
        }else if (strcmp(str,"7e4")==0){
                escape(buffer);

                memset(strGroupID,'\0',3);
                memcpy(strGroupID,buffer+10,2);
                intGroupID=strtol(strGroupID,NULL,16); // Hex

                memset(strPacketID,'\0',3);
                memcpy(strPacketID,buffer+16,2);
                intPacketID=strtol(strPacketID,NULL,16);

                memset(strMoteID,'\0',3);
                memcpy(strMoteID,buffer+18,2);
                intMoteID=strtol(strMoteID,NULL,16);

                memset(strReserved,'\0',3);
                memcpy(strReserved,buffer+20,2);
                intReserved=strtol(strReserved,NULL,16);

                for(i=0; i<MAX_CH; i++){
```

```
                                        memset(strCh,'\0',5);
                                        memcpy(strCh,buffer+22+i*4,4);
                                        interchange(strCh);
                                        intCh[i] = strtol(strCh,NULL,16);
                                }
                                //printf("Processing... [head]=%d,
[tail]=%d\n",head,tail); //test
                                //process the record based on the design format
                                record[tail][0] = day;
                                record[tail][1] = time;
                                if (intPacketID == 2){
                                        record[tail][2]=200+intReserved;
                                }else if (intPacketID == 3){
                                        record[tail][2]=300+intReserved;
                                }else{
                                        record[tail][2]=intGroupID;
                                }
                                record[tail][3] = intMoteID;
                                for (i=0; i<MAX_CH; i++){
                                        record[tail][4+i]=intCh[i];
                                }
                                //crc
                                strcpy(crcString,"");
                                for(i=0; i<17; i++){
                                        sprintf(crctemp, "%5.0f", record[tail][i]);
                                        tmp = trim_nocopy(crctemp);
                                        strcat(crcString, tmp);
                                }
                                record[tail][17] = crcFast(crcString,
strlen(crcString));
                                tail = (tail + 1)%S_QUEUE_LEN;
                        }
                }//end of while loop
                fflush(f1);
                fclose(f1);
                // Update semaphore value, set to 0
                if (!(f2 = fopen(semaphore, "w"))) {
                        perror ("Couldn't open semaphore file. \n");
                        exit(1);
                }
                smph[0] = '0';
                fwrite(smph,strlen(smph),1,f2);
                //printf("Data process done. Waiting for the next data...\n"); //test
                //printf("Post-semaphore is %s\n", smph); //test
                fflush(f2);
                fclose(f2);
```

```
                }else {
                        if (tail==head){ // If Storing Queue is empty, sync time
                                if (timecount>=1728){//sync time about every 12 hours
                                        printf("\n$4000 #\n"); // see CR206 program
                                        system("../home/xbow/apps/prelogger/syndate.sh");
                                        timecount=0;
                                        sleep(15);
                                }
                        }else if ((tail-head)==S_QUEUE_LEN-1||(head-tail)==1){//check
if storing queue is full
                                tail = head;
                                head = (tail + 1)%S_QUEUE_LEN;
                                //test
                                //printf("Queue Full... [head]=%d, [tail]=%d\n",head,tail);
                                sleep(15);
                        }else if ((tail-head)>=2||head>tail){
                                // Assemble the ready queue
                                readyqueue[0] = PakBusAdd;
                                readyqueue[1] = seqnum++;
                                if (seqnum == 65535) seqnum = 0;
                                for (i=0; i<36; i++){
                                        readyqueue[i+2] =
record[(head+i/18)%S_QUEUE_LEN][i%18];
                                }
                                readyqueue[38] = 0; // Reserved for Gateway Battery
                                readyqueue[39] = 0; // Reserved for Repeater Battery
                                result = 0;
                                rep = 0;
                                do{
                                        printf("$");
                                        for (i=0; i<40; i++){
                                                printf("%f ", readyqueue[i]);
                                        }
                                        printf("#\n");
                                        result=getResponse(tout);
                                }while (result==0 && rep++<2);
                                head = (head + 2)%S_QUEUE_LEN;
                                //if (head>S_QUEUE_LEN-1) head=0;
                                timecount++;
                                //printf("After processing, [head]=%d, [tail]=%d\n", head,
tail); //test
                        }
                }
        }
        }//end of endless loop
}//main
```

```
int getResponse(int n){
	char re[2];
	int res = 0;
	int error = 0;

	fd_set read_file_descr;
	struct timeval timeout;

	FD_ZERO(&read_file_descr);
	FD_SET(fileno(stdin), &read_file_descr);
	timeout.tv_sec = n;
	timeout.tv_usec = 0;

	error = select(1, &read_file_descr, 0, 0, &timeout);
	if (error == 0){
		// printf("Timeout.\n");
		return res;
	}else if (error < 0){
		// printf("Error.\n");
		return res;
	}else {
		memset(re, '\0', 2);
		scanf("%s", re);
		// printf("re=%s\n",re);
		if (strcmp(re,"1")==0){
			res = 1;
		}else if (strcmp(re,"2")==0){
			res = 2;
		}
	}
	return res;
}


escape(char strLine[]){//get rid of escape chars in the data
	char ch1[2];
	char ch2[2];
	char strBlk[3];
	char NewLine[256];
	int bhead = 0;
	int i,j;

	memset (NewLine, '\0', 256);
	for (i=0;i<strlen(strLine);i=i+2){
```

```
                strcpy(ch1,"a\0");
                memmove(ch1,strLine+i,1); // target: ch1, strLine+i: source, number to be moved:
1
                strcpy(strBlk,ch1);

                if(i+1<strlen(strLine)){
                        strcpy(ch2,"a\0");
                        memmove(ch2,strLine+i+1,1);
                        strcat(strBlk,ch2);

                        if (strcmp(strBlk,"7d")==0){
                                bhead=1;
                        }else if(bhead==1 && strcmp(strBlk,"5e")==0){//7d5e=>7e
                                strcpy(strBlk,"7e");
                                memset(NewLine+strlen(NewLine)-2,'\0',2);
                                bhead=0;
                        }else if(bhead==1 && strcmp(strBlk,"5d")==0){//7d5d=>7d
                                strcpy(strBlk,"");
                                bhead=0;
                        }else{
                                bhead=0;
                        }
                }else {
                        strcpy(strBlk,ch1);
                }
                strcat(NewLine,strBlk);
        }
        strcpy(strLine,NewLine);
        return;
}

interchange(char str[]){//put high bits and low bits to their right position
        char strtemp[5];
        memset(strtemp, '\0',5);
        memcpy(strtemp, str+2,2);
        strncat(strtemp,str,2);
        strcpy(str, strtemp);
        return;
}

int JulianDate (int y,int m ,int d){//Convert Julian date to normal date
        int days;
        if ((y % 4)==0){//check for leap year
                switch(m){
                case 1:
                        days=d;
```

```
                break;
        case 2:
                days=d+31;
                break;
        case 3:
                days=d+60;
                break;
        case 4:
                days=d+91;
                break;
        case 5:
                days=d+121;
                break;
        case 6:
                days=d+152;
                break;
        case 7:
                days=d+182;
                break;
        case 8:
                days=d+213;
                break;
        case 9:
                days=d+244;
                break;
        case 10:
                days=d+274;
                break;
        case 11:
                days=d+305;
                break;
        case 12:
                days=d+335;
        }
}else{
        switch(m){
        case 1:
                days=d;
                break;
        case 2:
                days=d+31;
                break;
        case 3:
                days=d+59;
                break;
        case 4:
```

```
                        days=d+90;
                        break;
                case 5:
                        days=d+120;
                        break;
                case 6:
                        days=d+151;
                        break;
                case 7:
                        days=d+181;
                        break;
                case 8:
                        days=d+212;
                        break;
                case 9:
                        days=d+243;
                        break;
                case 10:
                        days=d+273;
                        break;
                case 11:
                        days=d+304;
                        break;
                case 12:
                        days=d+334;
                }
        }
        return days;
}

char *trim_nocopy(char *s){
        char *start = s;
        char *end;
        // skip spaces at start
        while(*start && isspace(*start))
                ++start;

        char *i = start;
        // iterate over the rest remebering last non-whitespace
        while(*i)
        {
                if( !isspace(*(i++)) )
                        end = i;
        }

        // white the terminating zero after last non-whitespace
```

```
        *end = 0;

        return start; //
}

int max_array(double a[], int num_elements){
        int i,j;
        double max=-32000.0;
        for (i=12; i<num_elements*2-12; i++)//do not seach the first 6 numbers and the last 6
numbers
        {
                //printf("a[%d]=%f\n",i, a[i]);
                if (a[i]>max)
                {
                        max=a[i];
                        j=i;
                }
        }
        return(j);
}
```

Appendix H - The R script to derive α and c in the exponential decay model

```
> x<-c(1,12.38,7.59,5.1,10.7,6.62)
> y<-c(0.06,19.35,17.09,13.17,16.92,17.13)
> cor(x,y)
[1] 0.8717391
> fit<-lm(log(y)~log(x))
> fit

Call:
lm(formula = log(y) ~ log(x))

Coefficients:
(Intercept)       log(x)
     -2.301        2.400

> a<-exp(-2.301)
> a
[1] 0.1001586
> b=2.400
> summary(fit)

Call:
lm(formula = log(y) ~ log(x))

Residuals:
   1       2       3       4       5       6
-0.5128 -0.7755  0.2746  0.9683 -0.5597  0.6051

Coefficients:
```

```
        Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.3006     0.7568  -3.040  0.03841 *
log(x)        2.4001     0.3927   6.111  0.00363 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7995 on 4 degrees of freedom
Multiple R-squared:  0.9033,    Adjusted R-squared:  0.8791
F-statistic: 37.35 on 1 and 4 DF,  p-value: 0.003629

> anova(fit)
Analysis of Variance Table

Response: log(y)
        Df  Sum Sq Mean Sq F value   Pr(>F)
log(x)     1 23.8721 23.8721   37.35 0.003629 **
Residuals  4  2.5566  0.6391
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```