ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
COLLEGE OF NATURAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Dam Safety Monitoring Using Wireless Sensor *Networks*

By: Salahadin Seid

Advisor: Mulugeta Libsie (PhD)

A THESIS SUBMITTED TO
THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN COMPUTER SCIENCE

February, 2013
Addis Ababa

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
COLLEGE OF NATURAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Dam Safety Monitoring Using Wireless Sensor Networks

By: Salahadin Seid

Advisor: Mulugeta Libsie (PhD)

APPROVED BY:

BOARD OF EXAMINERS:

1. **Dr. Mulugeta Libsie, Advisor** _____
2. _____ _____
3. _____ _____

Acknowledgement

First of all, I would like to say "***Alhamdulilahi Rabilalemin***". Second, I would like to sincerely thank my advisor, Dr. Mulugeta Libse, for his supervision, advice, patience, and moral support throughout this thesis work. I taught a lot of bundled experience from his critical observation regarding this work and his incredible encouragement and support made me a better student in spite of myself.

I also thank Andualem Shigute (MSC student Water Engineering at Addis Ababa University) and Ismael Seid (Civil Engineer at Ethiopia road authority) for giving valuable ideas and guidance from civil engineering perspective for this thesis.

I would also thank all faculty members and colleagues in the department of Computer Science, Addis Ababa University especially Abubeker Yimam, Hailay Beyene, and Samuel Sisay and also staff of Computer Science Jimma University, for encouraging me in all aspect of this thesis.

Finally, my gratitude goes to my family (Seid Mussa, Neima Seid, and Fozia Seid) for their love, support and encouragement during all my studies and to my sweet heart, Nefissa Abdela, for sharing my deepest moments of joy and support.

Dedication

To: My Father (Seid Mussa Mohammed)

Table of Content

List of Tables

List of Figures

List of Acronyms

CH	Cluster Head
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
CTP	Collection Tree Protocol
DSM	Dam Safety Monitoring
ETX	Expected Transmission
FFT	Fast Fourier Transform
GSM	Global System for Mobile communication
HEED	Hybrid, Energy-Efficient, Distributed clustering protocol
LEACH	Low Energy Adaptive Clustering Hierarchy
MAC	Medium Access Control
MEMS	Micro Electro-Mechanical System
MHCRP	Multi-Hop Cluster based Routing Protocol
PDR	Packet Delivery Ratio
RF	Radio-Frequencies
SF	Serial Forwarder
SHM	Structural Health Monitoring
SMS	Short Message Service
SOC	System On Chips
TDMA	Time Division Multiple Access
TinyOS	Tiny Operating System
TOSSIM	TinyOS SIMulator
TTL	Time To Live
Wi-Fi	Wireless Fidelity
WSNs	Wireless Sensor Networks

Abstract

The rapid advances in processor, memory, and radio technology have enabled the development of distributed networks of sensor nodes capable of sensing and communicating using wireless media. This enables to develop novel applications that change the way we interact with the world around us.

In this thesis, we aim to develop a wireless sensor network application for dam safety monitoring application where the sensor network encompasses heterogeneous types of sensors distributed in different portion of a dam to gather data about the integrity of the dam. The sensor nodes are organized in multi-hop cluster based tree to sample the different physical parameters of the dam and route the reading to a central place for data management and visualization purpose. We have implemented a dam safety monitoring application and its application specific routing protocol on top of TinyOS operating system using nesC programming. We have also implemented data management and visualization methods to present the data to the user in a user-friendly manner and providing exact detail. For such purpose the following tools are developed: LiveWatch which is a Java application for data logging, a web application to visualize the information on the web, and SMS module for alarm notification. The application allows real-time collection and processing of sensor data and presenting to the user through an appropriate interface.

We have evaluated the performance of our application through simulation using TOSSIM simulator on MicaZ sensor nodes. Finally, we present some numerical results that demonstrate the feasibility of our application and routing protocol. The experiment showed encouraging results.

Keywords:

Wireless Sensor Networks, Dam Safety Monitoring, Sensors, Distributed System, Routing Protocol, Simulation

CHAPTER ONE: INTRODUCTION

1.1 Background

The emerging field of wireless sensor networks combines sensing, computation, and communication into a single tiny device. This integration opens the way to a surplus of new applications.

Wireless sensors have been developed for different applications such as military, environmental and habitat monitoring, medical health monitoring, structural health monitoring, etc.

Dam is a critical civil infrastructure for many nations. For instance, Ethiopia has constructed a number of dams and continues constructing for the purpose of generating hydroelectric power and for irrigation. As it is a critical civil infrastructure for the development of the country, an automated method to keep track of dam safety by obtaining useful information about the behaviors of the dams, predict the future values, and detect abnormalities (irregularities) as early as possible to enable a timely response is thus of utmost importance.

Dam failures are most likely to happen for one of the following reasons [1]. The first reason is due to overtopping which is caused by water spilling over the top of a dam which accounts for approximately 34% of all dam failures. The second reason is cracking which is caused by movements like the natural settling of a dam which accounts for 30% of all dam failures. The third reason for dam failure has been caused by piping (internal erosion caused by seepage). The remaining percentage of causes of dam failures includes structural failure of the materials used in dam construction and inadequate maintenance.

Traditionally Dam Safety Monitoring (DSM) is performed by examining the health of the existing dam by a team of persons who are qualified either by experience or education to evaluate a particular structure. It is based on a review of existing data and information, first hand input from field and operational personnel, and site inspection. This is a labor intensive activity and may reduce the quality of the data gathered. As a result, the reliability and accuracy of the dam safety monitoring task will be doubtful.

To monitor the safety of a dam in real time, it is necessary to adopt an automated monitoring system which increases dam monitoring frequency, decreases monitoring cost, reduces the

probability of catastrophic failures, save lives, property, water storage and help for continuous monitoring of the dam in real time and predict the behavior of the dam more accurately.

The task of dam safety monitoring includes measuring stress, strain, temperature, vibrating, cracks, osmotic pressure, bed rock deformation, dam horizontal displacement, vertical displacement, tilt, deflection, landslide, infiltration flow, uplift pressure, water flow pattern and onsite inspection. So as to measure this data, there are different types of sensors like resistance-type sensors, vibrating-wire sensors, temperature sensors, tensile wire sensors, static water level sensors, and vertical coordinate detector [9].

1.2 Statement of the Problem

A reliable dam safety monitoring system using a wireless sensor network should consist of wireless sensor nodes which are highly energy and processing power constraint devices, a technique for data collection, routing, energy efficient data processing and data management, and appropriate safety evaluation techniques to take appropriate action.

As will be discussed in the review of related work section, there have been a number of research works that focus on the techniques like data collection and management, event detection and localization, routing and data dissemination, and in-network processing/data aggregation of wireless sensor networks. There are also some recent works on finding mathematical techniques for dam safety evaluation from dam engineering perspective. And these previous works do not model the general framework of dam safety monitoring tasks. Thus in this thesis we integrates the techniques that are used in a wireless sensor network with the data management, safety analysis, and visualization techniques and this leads us to propose a model/framework for a system to perform automated safety monitoring of a dam using a wireless sensor network. Hence in this research, we will address the following research questions:

- Application specific data collection and sampling techniques considering the energy limitation of the wireless sensor nodes to reduce labor of dam safety personnel and to increase the quality of data and accuracy.
- Model routing and clustering protocols considering application specific requirements of dam safety monitoring application to avoid cabling which is common in existing dam safety monitoring instrumentations.
- Implementation of real time data management, visualization, and SMS alert techniques for timely

response and safe life and property.

1.3 Objectives

General Objective

The general objective of this thesis is to propose a general framework for dam safety monitoring using a wireless sensor network.

Specific Objectives

The specific objectives of this thesis include:

- Model the framework of dam safety monitoring system using a wireless sensor network.
- Simulate a wireless sensor node.
- Develop a simulation model which is simulating the network topology and protocols for real-time communication.
- Show the effectiveness of the wireless sensor network of dam safety monitoring system and compare with the conventional dam safety monitoring system.
- Model and develop data management, analysis, and visualization techniques.
- Performance evaluation through simulation to assess the performance of our application.

1.4 Scope and Limitations

This thesis work focuses to design and implement a wireless sensor network application for dam safety monitoring using simulator due to unavailability of real wireless sensor node in a local market. The main plan is to identify the requirements of dam safety monitoring tasks for a wireless sensor network application, to design data collection techniques using wireless sensor nodes, applying appropriate sampling techniques in data collection, to design application specific routing protocol, visualization, and alarm notification tools.

This thesis does not cover:

- Data compression techniques to overcome the bandwidth limitation of wireless sensor networks.

- Actual deployment on the dam site to see its practical operation

1.5 Methodology and Tools

To accomplish the objectives of this proposed thesis work, the following steps are followed:

1.5.1 Literature review

We will review published related works and also literatures to gather information about the study area relevant to this research work.

1.5.2 Interview and onsite observation

In order to design an efficient dam safety monitoring system based on a wireless sensor network, it is a must to understand the critical parameters and design requirements of a dam safety monitoring system. We have achieved this task by consulting a Water Engineer from the Civil Engineering Department of Addis Ababa University and a Civil Engineer from Ethiopian Road Authority.

As a result of analysis of dam safety monitoring system requirement, the framework has been modeled.

1.5.3 Software and hardware tools

Software and hardware tools, those that are relevant for development and simulation, will be identified and studied.

TinyOS – is a free and open source component based operating system and platform targeting Wireless Sensor Networks (WSNs) and it is an embedded operating system written in the nesC programming language. It is an operating system environment designed to run on embedded devices used in distributed wireless sensor networks [2].

nesC - is a component based, event driven programming language used to build applications for the TinyOS platform. It is built as an extension to the C programming language with components "wired" together to run applications on TinyOS [3].

TOSSIM - is a discrete event simulator for TinyOS applications. It helps to debug, test and analyze the behavior of the application on a desktop computer [4].

MicaZ - is a 3rd generation, tiny, wireless platform for smart sensors and it provides a wireless

communication with every node as router. It also has a connector for light, temperature, RH, pressure, acceleration/seismic, acoustic, magnetic and other MEMSIC Sensor Boards [6].

Transceiver - For wireless communication TI CC2420 RF transceiver [7] will be used. It is a chip 2.4 GHz IEEE 802.15.4 [8] compliant RF transceiver.

1.5.4 Simulation

Wireless sensor node simulation design

This is the fundamental objective of wireless sensor network based dam safety monitoring system. To accomplish this task there are two alternatives. The first alternative is designing the sensor node or use on-the-shelf sensor. The second alternative is to model the wireless sensor node using simulation tools. Currently, since it is not possible to get (buy) the wireless sensor node (the hardware) in the local market, we used the second alternative.

Explore Communication and synchronization protocol

Since the wireless sensor nodes used in the application need to communicate and synchronize with each other, they have to support IEEE 802.15.4/Zigbee implementation. So there is a need to explore the network topology and protocols.

1.5.5 Performance evaluation

After the system is implemented, its proper functioning is checked using the simulation tools with different values of parameters to be tested. In order to validate the capability and effectiveness of the designed wireless sensor network based dam safety monitoring system, there must be an evaluation metrics. Hill [10] explored the evaluation metrics that will be used to evaluate the wireless sensor network and also to evaluate the performance of individual nodes.

Evaluation metrics are dependent on the high-level objective of the application. Thus, based on our application objective, we have used the evaluation metric that are listed in Table 1.1 for our application.

Table 1.1: Evaluation metric for dam safety monitoring wireless sensor network

No.	Evaluation metric	Description
1.	Packet delivery ratio	It is the ratio of number of packets received at the destination to the number of packets sent by the source.
2.	Average power consumption	It is the average of consumed energy among all the nodes in the network.
3.	Latency	It is performance metric is used to measure the average end-to-end delay of data packet transmission.
4.	Sample Rate	This performance metric is used to measure the rate at which sensor nodes make sample and communicated to sink node.

1.6 Outlines of the Thesis

This thesis work comprises of seven chapters. The next Chapter covers the study of a WSN, the architecture, the protocols and applications. In Chapter three we deal with dam, types of dams, dams in Ethiopia, and dam safety monitoring techniques. Chapter four reviews related works. Chapter five deals with the main design issues and Chapter six discusses the development and simulation tools that are used for this work and also evaluation of the thesis and its result is presented. Finally, Chapter seven summarizes the thesis by proposing the recommendations and future works.

CHAPTER TWO: WIRELESS SENSOR NETWORKS

2.1 Introduction

Wireless sensor networks are a new class of distributed systems [14] where a collection of nodes are organized in a cooperative network intended to monitor the physical condition and to cooperatively pass the data through an ad-hoc network to the main location. Each node consists of a processing unit, multiple types of memory (program, data and flash memory), communication unit that enables to send and receive data with other nodes, power supply (commonly battery), and various sensors and actuators.

The development of wireless sensors was motivated by military applications such as battle field surveillance. Today such networks are deployed and continue to be deployed at an accelerated rate for different applications such as environmental monitoring and structural health monitoring. Thus, this emerging technology has the capacity to revolutionize the way we live and work in the 21st century [14].

Following this introduction, the chapter introduces the basic background and components of wireless sensor networks which are related to this thesis work. In section 2.2 the basic components of wireless sensor nodes are described. Section 2.3 presents the sensor network and the different protocols along with their design issue. Section 2.4 describes the basics of data aggregation and its advantage for wireless sensor applications. Section 2.5 contains a detailed description of the TinyOS Operating System. In Section 2.6, we describe the application of wireless sensor networks. Finally Section 2.7 presents a summary of the chapter.

2.2 Wireless Senor Node

A sensor node consists of computation, sensing and communication units. Building a wireless sensor network first of all requires the constituting nodes to be developed and made available. These nodes have to meet the requirements that come from the specific requirements of a given application: they might have to be small, cheap, or energy efficient, they have to be equipped with the right sensors, the necessary computation and memory resources, and they need adequate communication facilities.

There are a number of wireless nodes available for use in wireless sensor network research and

development. Some examples of such wireless sensor nodes are Mica mote family, EYES nodes, BTnodes, and ScatterWeb. Typical wireless sensor node components are shown in Figure 2.1 [23].

When choosing the hardware components for a wireless sensor node, evidently the application's requirements play a decisive factor with regard mostly to size, cost, and energy consumption of the nodes – communication and computation facilities as such are often considered to be of acceptable quality.

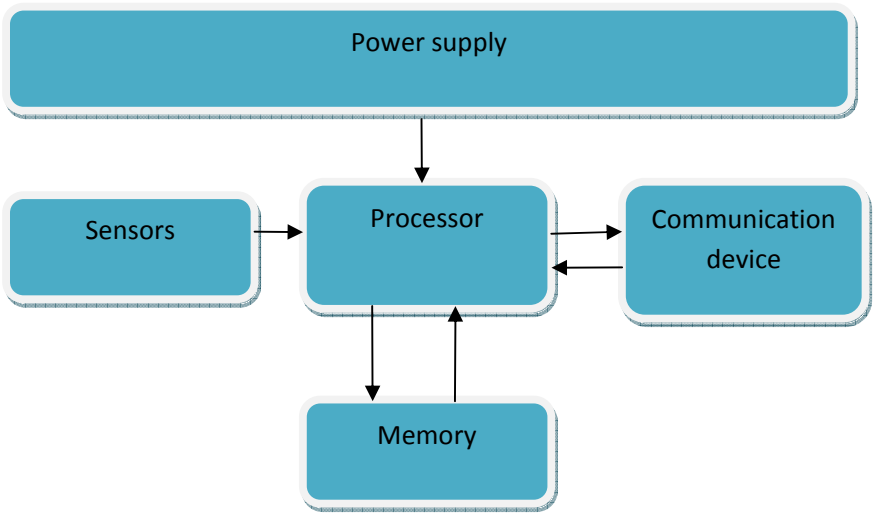


Figure 2.1: Typical sensor node components

The hardware basis of WSNs is driven by advances in several technologies. First, the advancement of system-on-chip (SOC) technologies. Second, the availabilities of RF circuits for short-distance communication. Thirdly, Micro-Electro-Mechanical System (MEMS) technology is now available to integrate a rich set of sensors onto the same chip.

The different components are described as follows [23].

Processor: is the core of a wireless sensor node which is in charge of processing data and executing the code that describes the operation of the sensor node. The processor gets data from the sensors, processes this data, decides when and where to send it, receives data from other sensor nodes, and decides on the actuator's behavior. It has to execute various programs, ranging from time-critical signal processing and communication protocols to application programs. System-on-chip (SOC) technology enables integrating a complete system on a single chip. Commercial SOC based embedded processors from Atmel, Intel, and Texas Instruments have

been used for sensor nodes such as UC Berkeley's nodes.

Memory: is used to store programs and data. Commonly random access memory (RAM) is used to store intermediate and temporary samples or packets from other nodes. Memories like Electrically Erasable Programmable Read Only Memory (EEPROM) are used to store the program code. However, the energy consumption by memory is an important factor for the appropriate design of the wireless sensor node.

Communication Device: is the hardware to enable the networking capability of the sensor nodes. Some usual methods for communication between the sensor nodes are Radio Frequencies (RF), optical communication, etc. RF is generally used because it provides a long range of transmission and reception at high rate with acceptable error rates for the required energy and also it does not require a direct line of sight between two neighbors.

Sensor: A sensor is an electronic component that measures the physical quantity and converts to the type that can be read by the user or other electronic device such as, for example, a microprocessor. Micro Electro-Mechanical System (MEMS) technology is now available to integrate a rich set of sensors onto the same chip. They are interfaces to the physical world which sense the environment parameter and convert them to a raw data (mostly voltage or current) for further processing within the processor. Commercially available sensors include thermal, acoustic/ultrasound, and seismic sensors, magnetic and electromagnetic sensors, optical transducer, chemical and biological transducer, accelerometer, and barometric pressure detectors. These sensors can be used in a broad range of applications. Commercially there are readily available sensors for different applications, for instance Table 2.1 lists sensors from Geokon [29] that can be used for dam safety monitoring application.

Table 2.1: Commercial Sensors

No.	Measurement	Sensor type
1.	Reservoir level	Geokon Model 4500AL vibrating wire pressure transducer
2.	Seepage	Geokon Model 4580-3V vibrating wire pressure
3.	Pore pressure/Uplift	Geokon Model 4500AL vibrating wire pressure transducer
4.	Surface settlement	Geokon Model 4600 Settlement System

Power Supply: For wireless sensor nodes, the power supply is a crucial system component. There exists a large quantity of power supply options for a sensor node. The most common option is the use of batteries; other options are scavenging energy from the environment where the sensor node is exposed, the most popular example is solar cell.

The integration of the above technologies has made it possible to integrate sensing, computing, communication, and power components in to tiny sensor nodes.

2.3 Wireless Networking

Wireless sensor networks have characteristics that are different from traditional wireless networks. The main characteristics of a WSN include its ability to cope with node failures, node's dynamic mobility and network topology, frequent communication failures, the presence of heterogeneous nodes, scalability to large scale deployment, ability to withstand harsh environment conditions, and node's severe power constraints [28]. This makes WSN an active research area to design different models and protocols. IEEE 802.15.4 [15] is the main standard technology for WSNs which intends to offer the fundamental network lower layers and focuses on low-cost and low-speed ubiquitous communication between devices. In this section, we review IEEE 802.15.4 standard which covers the Physical and Medium Access Control (MAC) layer of Wireless Sensor Networks and the issues of routing protocols for wireless sensor networks. In Addition, we also discuss the Collection Tree Protocol (CTP) and different clustering protocols since we customized to design application specific routing protocol for dam safety monitoring application.

2.3.1 Physical Layer Protocols

The physical layer (PHY) defines the physical and electrical characteristics of the network, for instance, specifying the receiver sensitivity and transmitting output power. The physical layer also specifies the raw data rate characteristics, which can either be selected to offer a larger coverage area or higher throughput.

The basic task of this layer is thus data transmission and reception at the physical/electrical level, which involves modulation and spreading techniques that map bits of information in such a way they can travel through the air. The PHY tasks can be summarized as follows:

1. Enable/disable the radio transceiver (since low duty cycle saves energy)

2. Compute Link Quality Indication for received packets
3. Energy Detection within the current channel by means of signal strengths estimation
4. Listen to channels and declare availability or not.

Many of the WSN systems today are based on IEEE 802.15.4 [15] protocols due to their low-power consumption. IEEE 802.15.4 standard covers the Physical and Medium Access Control (MAC) layer of Wireless Sensor Networks.

The 802.15.4 physical layer specifies two different services: The PHY data service which controls the transmission and reception of the PHY Protocol Data Units (PPDUs) and the PHY management service performs Energy Detection (ED) in the channel and also performs Clear Channel Assessment (CCA) before sending the messages and provides Link Quality Indication (LQI) for the received packets.

2.3.2 Medium Access Layer Protocols for WSN

A Medium Access Control (MAC) protocol is the first protocol layer above the Physical Layer (PHY). Its fundamental task is to regulate the access of a number of nodes to a shared medium in such a way that certain application-dependent performance requirements are satisfied.

The IEEE 802.11 protocol was the first standard for wireless local area networks (WLAN) [27]. Even though many efforts have been made to use 802.11 for WSNs, the high power consumption and excessively high data rate of IEEE 802.11 protocol are not suitable for WSNs. This fact has motivated several research efforts to design energy efficient MAC protocols.

The authors in [24] made a survey on the different MAC protocols including Sensor-MAC (S-MAC), WiseMAC, Spatial TDMA and CDMA, Traffic-Adaptive MAC protocol (TRAMA), Node Activation Multiple Access (NAMA), etc. and presented the advantages and limitations of each MAC protocol.

The IEEE 802.15.4 MAC Standard provides information about type and association of devices, channel access mechanism, packet delivery, frame structure, guaranteed packet delivery, possible network topologies and security issues. In order to communicate with the upper layers, it provides the MAC data service and the MAC management service. The MAC data service enables transmission of MAC Protocol Data Units (MPDU) across the PHY data service. The

MAC sub layer features include beacon management, channel access, frame validation, acknowledged frame delivery, association and disassociation. The MAC also provides support for implementing defined security mechanisms like Data Encryption, Frame Integrity and Sequential Freshness [8].

2.3.3 Routing Layer Protocols for WSN

In this subsection, we will describe the design issues and challenges of routing protocols and also the basic operations of CTP routing protocol, focusing on the details that are relevant to this thesis.

In wireless sensor applications like habitat monitoring that require the deployment of large amount of nodes, to transfer the packet that is sensed by sensor nodes to the base station, intermediate nodes are required to relay packets for the successful delivery of the packet to the base station. Otherwise the packet will be lost due to low transmission range of the sensor nodes. Such an intermediate node has to decide to which neighbor to forward the incoming packet which is not destined for it. Such decision is made by the routing protocol which runs on every node in the network.

There exist different routing protocols for wired and also for ad-hoc networks. However they are hardly applicable for wireless sensor networks due to their high power consumption. They are also designed to support general routing request in wireless networks, without considering specific communication pattern in WSN. Customization of these protocols for WSN and the developments of new routing techniques are needed, even though it is very challenging due to the inherent characteristics that distinguish these networks from other wireless networks like mobile ad-hoc networks or cellular networks. Some of the routing challenges and design issues that affect the routing process in WSNs are [50]:

- *Node Deployment:* Node deployment in WSNs is application dependent and affects the performance of the routing protocol. The deployment can be either deterministic or randomized. In deterministic deployment, the sensors are manually placed and data is routed through pre-determined paths. However, in random node deployment, the sensor nodes are scattered randomly creating an infrastructure in an ad-hoc manner. If the resultant distribution of nodes is not uniform, optimal clustering becomes necessary to

allow connectivity and enable energy efficient network operation. In our dam safety monitoring application, the sensor nodes are deployed in deterministic manner where nodes are deployed in the critical section of the dam.

- *Energy Consumption without Losing Accuracy:* sensor nodes can use their limited supply of energy for performing computations and transmitting information in a wireless environment. Thus, energy conserving forms of communication and computation are essential. The malfunctioning of some sensor nodes due to power failure can cause significant topological changes and might require rerouting of packets and reorganization of the network. Thus, the routing protocol designed for an application where the sensor nodes get power supply from battery and left unchecked for long time, as the case of our proposed application, should consider the limed energy supply of the sensor nodes.

- *Data Reporting Model:* Data sensing and reporting in WSNs is dependent on the application and the time criticality of the data reporting. Data reporting can be categorized as either time-driven (continuous), event-driven, query-driven, or hybrid. The time-driven delivery model is suitable for applications that require periodic data monitoring. As such, sensor nodes will periodically switch on their sensors and transmitters, sense the environment and transmit the data of interest at constant periodic time intervals. In event-driven and query-driven models, sensor nodes react immediately to sudden and drastic changes in the value of a sensed attribute due to the occurrence of a certain event or a query is generated by the base station. A combination of the previous models is also possible. In our proposed application, we modeled adaptive time driven data reporting model as it will be described in detail in Chapter 5.

- *Node/Link Heterogeneity:* In many studies, all sensor nodes were assumed to be homogeneous, i.e., having equal capacity in terms of computation, communication, and power. However, depending on the application a sensor node can have different role or capability. The existence of heterogeneous set of sensors raises many technical issues related to data routing. In our proposed dam safety monitoring application, there are diverse mixtures of sensor nodes which include water level sensor, uplift pressure sensor, settlement sensor, and seepage sensor. We considered the availability of heterogeneous

sensor nodes in our monitoring application while we design the routing protocol as is described in Chapter 5.

- *Fault Tolerance:* Some sensor nodes may fail or be blocked due to lack of power, physical damage, or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network. If many nodes fail, MAC and routing protocols must accommodate formation of new links and routes to the data collection base stations.

- *Scalability:* The number of sensor nodes deployed in the sensing area may be in the order of hundreds or thousands, or more. Any routing scheme must be able to work with this huge number of sensor nodes. In our case, the proposed routing protocol should scale for monitoring an application either for small or large dam.

- *Network Dynamics:* Most of the network architectures assume that sensor nodes are stationary, for example, the case of our dam safety monitoring application. However, mobility of either the base station or sensor nodes is sometimes necessary in many applications. Routing messages from or to moving nodes is more challenging since route stability becomes an important issue, in addition to energy, bandwidth, etc.

- *Coverage:* In WSNs, each sensor node obtains a certain view of the environment. A given sensor's view of the environment is limited both in range and in accuracy; it can only cover a limited physical area of the environment. Hence, area coverage is also an important design parameter in WSNs.

- *Data Aggregation:* Since sensor nodes may generate significant redundant data, similar packets from multiple nodes can be aggregated so that the number of transmissions is reduced. Data aggregation is the combination of data from different sources according to a certain aggregation function, e.g., duplicate suppression, minima, maxima and average. This technique has been used to achieve energy efficiency and data transfer optimization in a number of routing protocols. Detail description about data aggregation is given in Section 2.4.

Following this, we describe the basics of Collection Tree Protocol (CTP) [41], focusing on the details relevant to this thesis. Then we will describe the two popular clustering techniques: Low Energy Adaptive Clustering Hierarchy (LEACH) [42] and Hybrid, Energy-Efficient, Distributed (HEED) clustering protocol [47] since we adapt them for our application specific protocol which is modeled and described in Chapter 5.

CTP is a tree based collection protocol that delivers data to the sink node. To create the tree, some nodes in a network advertise themselves as tree roots then other nodes which hear the advertisement beacon join the network by making the root nodes as parent. Similarly the children of the root nodes again advertize to other nodes to join the network. To accomplish these tasks the three major components of CTP interact to maintain the routing tree, routing packet and determining the link quality between the nodes. The three components of CTP are Routing Engine (RE), Forwarding Engine (FE), and Link Estimator (LE). Following this, we discuss the basic operations of these components.

Routing Engine: The Routing Engine, an instance of which runs on each node, takes care of sending and receiving beacons as well as creating and updating the routing table. This table holds a list of neighbors from which the node can select its parent in the routing tree. The table is filled using the information extracted from the beacons. Along with the identifier of the neighboring nodes, the routing table holds further information, like a metric indicating the quality of a node as a potential parent. The metric that CTP used to measure the quality of a node to be a parent is the ETX (Expected Transmissions), which is communicated by a node to its neighbors through beacons exchange. This metric is computed by the LE component of CTP.

Generally RE, performs the following operations [41]:

- **Sending beacons:** A node sends a beacon to send a routing message to the nodes. CTP incorporates adaptive beacon techniques using the Trickle algorithm [40] to control the rate of sending a beacon instead of sending beacon periodically. Using Trickle, each node progressively reduces the sending rate of the beacons so as to save energy and bandwidth. A beacon is sent at a random instant within a given time interval, whose minimum length $I^{min}{}_{b}$ is set a priori. The length $I^{min}b$ of the interval is doubled after each transmission so that the frequency at which beacons are sent is progressively reduced. In order to avoid a too long absence of beacon transmissions, however, a maximal length of the sending

interval, which we refer to as $I^{max}_{b,}$ is fixed a priori. The occurrence of specific events can cause the value of I^{min}_{b} to be reset. These events include: the detection of a routing loop, a congested node.

- **Routing table updates:** The RE takes care of updating the routing table by retrieving information about available neighbors, their selected parent, congestion status and multi-hop ETX. The routing table is updated asynchronously upon reception of routing frames. It keeps only the most reliable neighbors by removing neighbors from the routing table which are not relevant.

Forwarding Engine: takes care of forwarding data packets which may either come from the application layer of the same node or from neighboring nodes. Generally FE has the following responsibilities [41]:

- ***Queuing Packet and retransmissions:*** The FE stores the data packets to send in a FIFO queue. To forward a packet, the FE first retrieves the identifier of the current parent from the RE and sends the data packet if the parent is not congested. After sending a packet the FE awaits for a corresponding acknowledgment before removing it from the head of the queue. If the acknowledgment is not received within a given time interval, the FE will try and retransmit it until a pre-specified maximum of retransmission attempts have been performed. After the maximal number of retransmission attempts has been reached, the packet is discarded.
- ***Setting congestion flag:*** If a node is chosen as parent from several neighbors, or if it must perform many retransmissions attempts, the queue of its FE may quickly fill up with unsent packets. Since this may eventually cause the node to drop further incoming packets the FE notifies this congestion status.
- ***Checking duplicate packets***: In order to detect duplicate packets, for each incoming packet it evaluates the origin parameters which specify the identifier of the node which sent the packet, the sequence number of the current frame, and the hop count of the packet. By comparing the value of an incoming packet with that of the packets stored in the forwarding queue, the FE can detect duplicates.

- ***Checking routing loop:*** The FE also has a mechanism to detect the occurrence of routing loops. To do this, the FE compares the (multi-hop) ETX of each incoming packet with the (multi-hop) ETX of the current node.

Link Estimator: The Link Estimator takes care of determining the inbound and outbound quality of 1-hop communication links. The metric that expresses the quality of a link is refereed to as Expected Transmission (ETX). The LE computes the 1-hop ETX by collecting statistics over the number of beacons received and the number of successfully transmitted data packets. The ETX of a node is defined as the ETX of its parent plus the ETX of its link to its parent [39].

How to compute the ETX: For each neighbor, the LE determines the 1-hop ETX considering the quality of both the ingoing and outgoing links.

The quality of the outgoing link is computed as follows. Let n_u be the number of unicast packets sent by the node (including retransmissions), n_a the corresponding number of received acknowledgments, and Q_u is quality of the outgoing link. LE computes the value of Q_u after a number of window of length w_u of unicast packets have been sent. The values of n_u and n_a are then reset and, after w_u transmission, a new value of Q_u is computed. The quality of the outgoing link is then computed as the ratio:

$$Q_u = \frac{nu}{na}$$

The quality of the ingoing one computed as follows. If n_b is the number of beacons received by a node from a specific neighbor and N_b is the total number of beacons broadcasted by the same neighbor, then the quality of the corresponding (ingoing) link is given by the ratio:

$$Q_b = \frac{n_b}{N_b}$$

As for Q_u, the values of Q_b are computed over a window of length w_b. Before being forwarded to the function that determines the 1-hop ETX, however, the value of Q_b is passed through an exponential smoothing filter. This filter averages the new value of Q_b and that of previous

samples but weighting the latter according to an exponentially decaying function. In mathematical notation, the *k*th sample of Q_b, indicated as $Q_b[k]$, is computed as:

$$Q_b[k] = \alpha_b \frac{n_b}{N_b} + (1 - \alpha_b) Q_b[k-1]$$

Where α_b is a smoothing constant that can take values between 0 and 1, default value is 0.9.

Each time a new value of Q_u or Q_b for a specific neighbor is available; the 1-hop ETX metric relative to the same neighbor is accordingly updated. Let Q be the new value of either Q_u or Q_b and ETX_{old} the previously computed value of the 1-hop ETX. The updated value of the 1-hop ETX is then computed as follows:

$$ETX = \alpha_{ETX} Q + (1 - \alpha_{ETX}) ETX_{old}$$

Where α_{ETX} is a smoothing constant, default value is 0.9.

Since the multi-hop ETX of a neighbor quantifies the expected number of transmissions required to deliver a packet to a sink using that neighbor as a relay, the node clearly selects the neighbor corresponding to the lowest multi-hop ETX as its parent. The value of this multi-hop ETX is then included by the node in its own beacons so as to enable lower level nodes to compute their own multi-hop ETX. The multi-hop ETX of a sink node is always 0. The comprehensive implementation detail of CTP protocol is presented in [41, 46].

CTP is a collection protocols where sensor nodes sample their readings and forward to the parent node then finally reached to the base station. However, some sensor network applications like dam safety monitoring, in our case, require data aggregation to be performed before the data is routed to the base station. For such purpose, for clustering sensor nodes and performing data aggregation operation on a cluster head, a clustering protocol is needed. Node clustering is also employed to further balance the load among sensor nodes and prolong the network lifetime. There are a number of clustering protocols in the literature, thus we discuss the popular clustering protocols below.

LEACH [42] is a self-organizing, adaptive clustering protocol that uses randomization to

distribute the energy load evenly among the sensor nodes in the network. This protocol assumes the sensor nodes and the base station are at fixed location and they are homogeneous and energy constrained. In this cluster protocol, the nodes organize themselves into local clusters, where one node acts as a cluster-head and then utilizes randomized rotation to rotate the position of the cluster head so as not to drain the battery of a single sensor. The decision to become a cluster-head depends on the amount of energy left at the node. Each node decides whether or not to become a cluster-head for the current round based on the suggestion percentage of the cluster heads for the network that is fixed a priori and the number of times the node has been a cluster-head so far. A node chooses a random number between 0 and 1 and compare with the threshold $T(n)$. If the random number is less than the threshold $T(n)$, the node becomes the cluster-head for the current round. The threshold $T(n)$ is determined using the equation:

$$T(n) = \begin{cases} \dfrac{p}{1 - p * (r \bmod \frac{1}{p})} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases}$$

Where p = the desired percentage of cluster heads, r = the current round and G is the set of nodes that have not been cluster heads in the last 1/p rounds. Then after the cluster heads are selected for the current round, they advertize messages for the rest of the nodes. The non- cluster nodes which hear the advertisement decide the cluster to which they belong based on the received signal strength for that round. Then it creates clusters where non-cluster nodes transmit their sampled data to its cluster head and all cluster heads transmit directly to the data sink. It performs one-hop communication, a communication between the node and elected cluster head and a communication between cluster head with the base station. Finally, it creates cluster topology where nodes can send packages directly to the cluster head and thereafter to the sink.

Because of this one-hop topology nature, this kind of cluster formation is not suitable for a network deployed in a large region, as the case of our dam safety monitoring application, where cluster head maybe can't reach the base station due to is too far away from it.

HEED [47], hybrid energy efficient distributed clustering approach for ad-hoc sensor networks is an algorithm that assumes the sensor nodes are sensor quasi-stationary and node location unaware, the links are symmetric, and energy consumption is non-uniform for all nodes. It

periodically selects cluster heads according to the sensor residual energy and a node proximity to its neighbors. Initially, HEED sets an initial percentage of cluster heads among all *n* nodes (C_{prob}) and its probability of becoming a cluster head, CH_{prob}, before a node starts executing HEED, is:

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

Where $E_{residual}$ is the estimated residual energy of the node, $E_{\max}$ is a reference maximum energy, and C_{prob} is a small constant fraction used to limit the number of initial cluster head announcements. CH_{prob} is not allowed to fall below a small probability, $p_{\min}$, to ensure constant time termination. During each iteration, a node arbitrates among the cluster head announcements it has received to select the lowest cost cluster head. If it has not received any announcements, it elects itself to become a cluster head with probability CH_{prob}. If successful, it sends an announcement indicating its "willingness" to become cluster head. The node then doubles its probability CH_{prob}, waits for a short iteration interval t_c, and begins the next iteration. A node stops this process one iteration after its CH_{prob} reaches 1.

If a node elects to become a cluster head, it sends an announcement message *cluster head msg (Node ID, selection status, cost),* where the selection status is set to tentative CH, if its CH_{prob} is less than 1, or final CH, if its CH_{prob} has reached 1. A node considers itself "covered" if it has heard from either a tentative CH or a final CH. If a node completes HEED execution without selecting a cluster head that is final CH, it considers itself uncovered, and announces itself to be a cluster head with state final CH. A tentative CH node can become a regular node at a later iteration if it finds a lower cost cluster head. Note that a node can elect to become a cluster head at consecutive clustering intervals if it has high residual energy and low cost.

2.4 Data Aggregation and Collaborative Signal Processing

In a typical wireless sensor network application, data is collected by the sensor nodes, and needs to be made available at some central base station, where it is processed, and analyzed.

Data generated by the sensor nodes can be jointly processed while being forwarded towards the

base station, which means by fusing together sensor reading related to the same event or physical quantity before transmitting to the base station. Thus, a technique called data aggregation deals with this distributed processing of data within the network. It has significant advantage on energy consumption and overall network efficiency because it reduces the number of transmissions [31].

Aggregation techniques require an effective aggregation function. There are several types of aggregation functions and most of them are closely related to the specific sensor application. Aggregation functions can compress and merge data according to either a lossy, where the original values cannot be recovered after having merged them by means of the aggregation function, or a lossless approach that allows compressing the data by preserving the original information. An aggregation function also needs to take into account the very limited processing and energy capabilities of sensor devices [31].

Collaborative signal processing algorithms are another enabling technology for WSNs. Since sensor reading is usually imprecise due to noise and interference from the environment, information fusion can be used to process data from multiple sensors in order to filter noise and provide more accurate interpretation of information generated by a large number of sensor nodes. A rich set of techniques are available in this context, including Kalman Filtering, Bayesian Inference, Neural Networks, and Fuzzy Logic [47].

In our proposed sensor application, however, we only consider to use a data aggregation operation in our proposed application specific routing protocol in Chapter 5.

2.5 Operating System

Sensor nodes are generally constrained in memory and computational capacity. The small physical size and the low cost intended for the devices impose a limit that must be taken into account when designing an application. Typical platforms are equipped with a code memory of dozen KB: for example the MicaZ platform is equipped with 128KB. These limitations motivate the development of specific oriented operating systems for this kind of hardware.

The traditional tasks of an operating system are controlling and protecting the access to resources (including support for input/output) and managing their allocation to different users as well as

the support for concurrent execution of several processes and communication between these processes. However, these tasks are required in an embedded system partially since the executing code is much more restricted and the systems do not have the required resource to support the full feature operating system as in a general-purpose system.

The characteristics of WSNs impose additional challenges on operating system design for WSN, and consequently, OS design for WSN deviates from traditional OS design. The operating system for wireless sensor networks should support the specific needs of these systems. The authors in [26] made a survey on different operating systems for wireless sensor networks with the focus of concerns such as architecture of the operating system, programming model of the operating system, scheduling techniques, memory management and protection, communication protocol support, resource sharing, and point out strengths and weaknesses of WSNs operating system such as TinyOS, Contiki, MANTIS, Nano-RK, and LiteOS keeping in mind the requirements of emerging WSN applications.

TinyOS [2] is a de facto standard operating system for sensor nodes. It is written using the nesC language [3] and provides an event driven operating system. The TinyOS component library includes network protocols, distributed services, sensor drivers, and data acquisition tools. TinyOS supports a wide range of hardware platforms and has been used on several generations of sensors nodes. Supported processors include the ATmel AT90L- series, ATmel ATmega-series, and Texas Instruments MSP-series of processors. TinyOS also supports RFM TR1000 and Chipcon CC1000 radios. Thus, TinyOS applications can be compiled to run on any of these platforms without modification of the source code. More importantly, TinyOS supports an extensive development environment that incorporates visualization, debugging, and support tools as well as a fine grained simulation environment.

A TinyOS application consists of a set of components and a scheduler. Components can be hardware abstractions, synthetic hardware, and high-level software. Each component is described by four elements: a set of commands, a set of events, a frame, and a set of tasks. All the commands, events, and tasks are executed in the context of the frame. The sets of commands and events can also be described as the component's interface to the rest of the system. Commands can be defined as non blocking requests to lower-level components to initiate an action; and they normally post a task for later execution. They can also initiate lower component commands, but

those have to be completed in a short period of time. However, they cannot initiate events. Events represent hardware events or the completion of commands. An event can store information to each frame, fire higher level events, call lower level commands, or post tasks. Finally, tasks are the component elements that perform the main job. They are able to call lower level commands, fire higher level events and post other tasks.

In order to allow concurrency, TinyOS provides a scheduler which allows scheduling tasks and preemption of tasks by events. Thus the desired concurrency inside a component is accomplished by the asynchronous execution of the events and tasks [51].

2.6 Application of Wireless Sensor Networks

The ultimate goal of wireless sensor network research is to enable novel applications that change the way we interact with the world around us [10]. Wireless sensor networks support a wide range of useful applications. In this section we identify some of these applications to show the current state of art of wireless sensor networks.

Environment Monitoring: is a key application area of wireless sensor networks. The large space coverage requirement involved in such applications requires large number of low cost sensor nodes that can be easily dispersed throughout the region. These applications include monitoring of micro-climate, detection of forest fire, flood, and monitoring of precision agriculture which is done by collecting real-time data on weather, soil and air quality, and crop maturity to make smarter decisions. Example of such application can be cited in [37].

Military Application: WSN technology has been used in monitoring inimical forces, friendly forces and equipment, targeting, battle field surveillance, battle damage assessment, nuclear, biological, and chemical attack detection. It replaces single high cost sensor assets with large arrays of distributed sensors making them robust and self organizing networks that are easily deployable by untrained troops in any situation. Example of such application can be cited in [16].

Civil Engineering Applications: Sensors can be used for civil engineering applications. A research has been done in recent years to develop sensor technology that is applicable for buildings, bridges, and other structures [22]. The goal is to develop ‘‘smart structures’’ that are able to self-diagnose potential problems and self-prioritize requisite repairs. This technology is attractive for earthquake-active zones. Although routine mild tremors may not cause visible

damage, they can give rise to hidden cracks that could eventually fail during a higher-magnitude quake. Smart dust nodes, tiny and inexpensive sensors developed by UC–Berkeley engineers, are promising in this regard. The battery-powered matchbox-sized nodes are designed to sense the dynamic response of the structure that helps for civil engineering analysis. Example of such applications can be cited in [13, 33].

2.7 Summary

To summarize, this chapter has provided an overview and description of the components of wireless sensor networks and applications. It began with the concept of the wireless sensor node and continued by introducing the different standards and protocols in the area of wireless sensor networks, their applications, related concerns and issues, and, finally, the application of wireless sensor networks in different areas. Its intention is to provide the theoretical background knowledge that is necessary for the design and modeling of dam safety monitoring application that is described in Chapter 5.

CHAPTER THREE: DAM SAFETY

3.1 Introduction

Water is one of our most important resources; nothing can live without it. But there has to be the right amount of water in the right place at the right time. Throughout the history of humankind, people have built dams to maximize the use of this vital resource. Today, dams are beneficial to communities and individuals for many reasons. Dams provide flood control, water supply for drinking, irrigation for farming, recreational areas, and clean, renewable energy through hydropower.

Dams provide a life-sustaining resource to people especially to countries like Ethiopia that have large water resources. They are an extremely important part of this nation's infrastructure—equal in importance to bridges, roads or other major elements of the infrastructure. Thus, the dam owner should assess the safety of the dam to extend the life span of the dam and prevent the risk associated with dam failure.

Following this introduction, this Chapter gives background introduction about dam and dam safety. Section 3.2 describes the common types of dams and Section 3.3 list currently available dams and their features in Ethiopia. Section 3.4 describes dam failure and lists recent major dam failure incidences in the world. Section 3.5 describes dam safety monitoring procedures and instruments that are currently in use by dam owners. Finally Section 3.6 summarizes the chapter by pointing out the problem of current dam safety monitoring techniques and indicating an alternative solution.

3.2 Dam Types

Dams are manmade barriers, constructed on natural terrain in order to control or store water [25]. There are different types of dams classified by the material and design used in construction. Based on the dam design, the different types of dams include embankment dam, concrete dam, and arch dam.

Embankment Dam - also termed as earth fill dam or rock fill dam, is the most common type of dam in use today. It has the general shape shown in Figure 3.1 [1]. Materials used for embankment dams include natural soil or rock, or waste materials obtained from mining or milling operations. The ability of an embankment dam to resist the reservoir water pressure is

primarily a result of the mass weight, type and strength of the materials from which the dam is made [1]. Such dam type in Ethiopia includes Gilgel Gibe I.

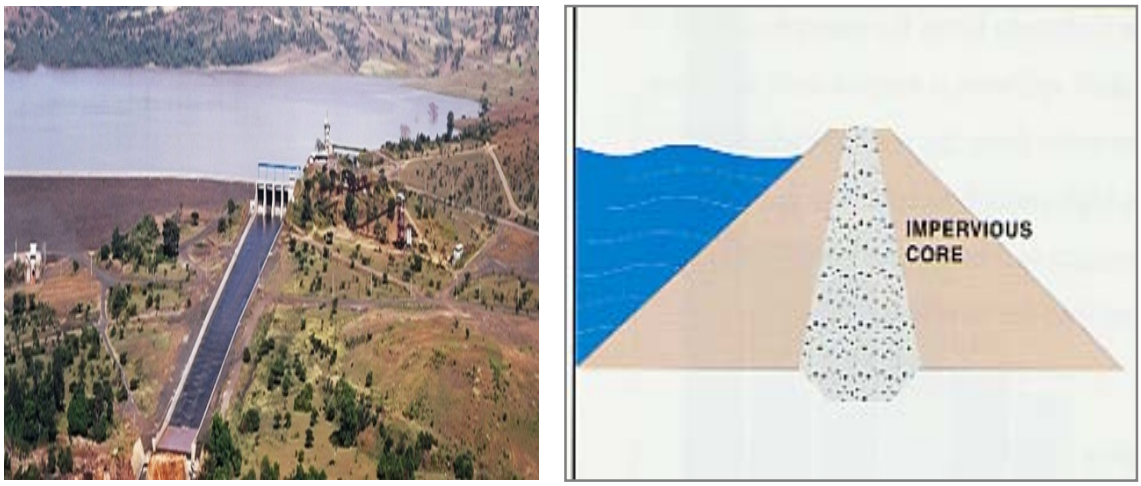


Figure 3.1: Embankment Dam

Concrete Dams - may be categorized into gravity and arch dams according to the designs used to resist the stress due to reservoir water pressure. Typical concrete gravity dams are shown in Figure 3.2 [1] and are the most common forms of concrete dams. The mass weight of concrete and friction resist the reservoir water pressure. Gravity dams are constructed of vertical blocks of concrete with flexible seals in the joints between the blocks [1]. Concrete dam in Ethiopia includes Gilgel Gibe III and the Grand Renaissance dam.

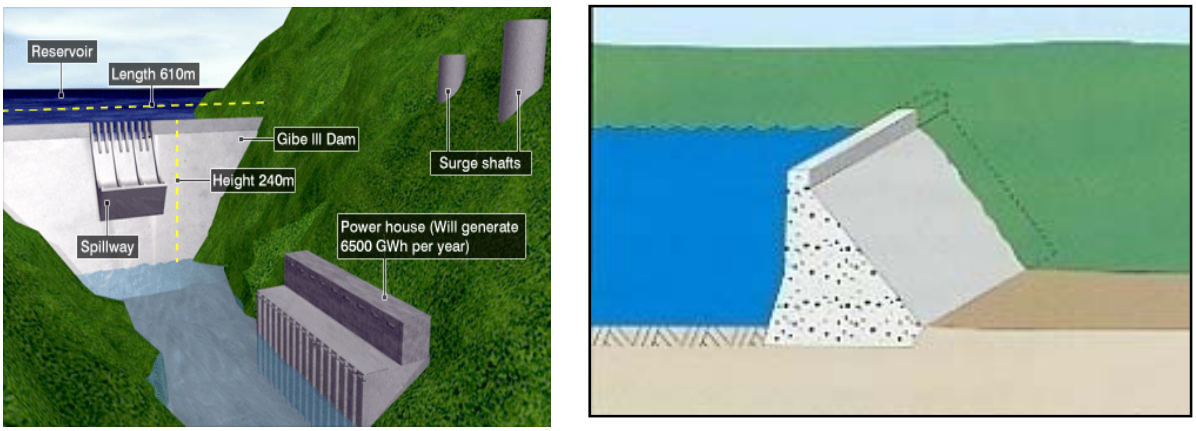


Figure 3.2: Concrete Dam

Arch Dams - the reservoir water forces acting on an arch dam are carried laterally into the abutments. The shape of the arch may resemble a segment of a circle or an ellipse, and the arch may be curved in the vertical plane as shown in Figure 3.3 [1]. Such dams are usually

constructed of a series of thin vertical blocks that are keyed together; barriers to stop water from flowing are provided between blocks. Variations of arch dams include multi-arch dams in which more than one curved section is used, and arch-gravity dams which combine some features of the two types of dams [1]. Typical example of Arch-gravity dam in Ethiopia is Tekeze dam.

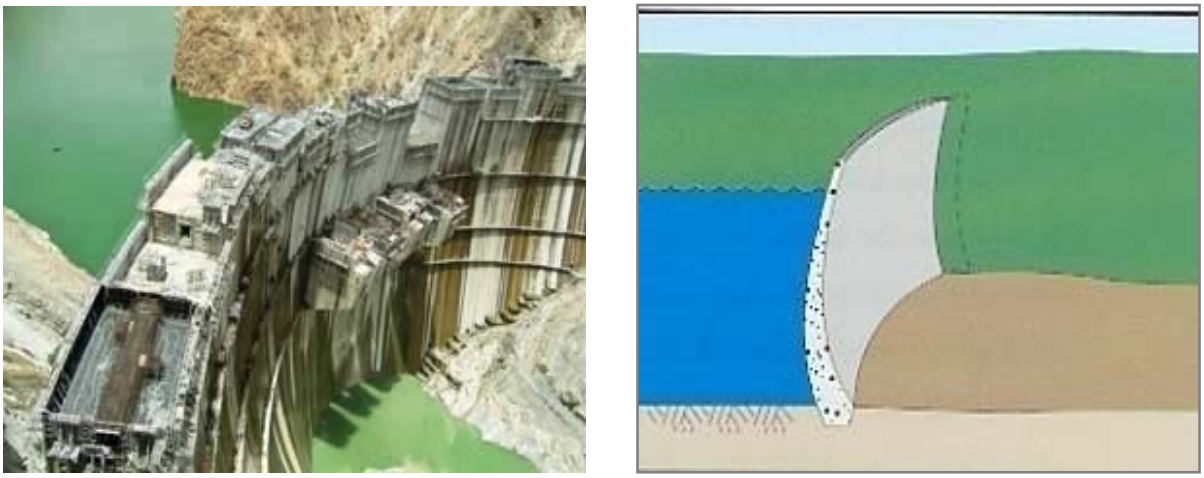


Figure 3.3: Arch Dam

3.3 Dams in Ethiopia

Ethiopia is a tower of water, a powerhouse of Africa due to its high hydropower potential. Even though a fraction of this potential has been exploited so far, the government has an ambitious dam building program. The benefits of the dams are not only limited to hydropower but also there are multi-purpose dams that are designed to provide water for irrigation like Beles and Fincha, and flood control. Table 3.1 [5] shows the list of major Ethiopian dams with their features.

Table 3.1: List of Ethiopian dams with their features

No.	Dam name	Installed capacity (Electricity)	Commissioned	Basin	Dam type	Height x length
1.	Fincha	134 MW	1973	Fincha Blue Nile	Embankment	20m x 340 m
2.	Gilgel Gibe I	180 MW	2004	Gilgel Gibe river	Embankment	40m x 704m
3.	Gilgel Gibe II	420 MW	2010	Omo River (no dam, fed by	No dam	---

No.	Dam name	Installed capacity (Electricity)	Commissioned	Basin	Dam type	Height x length
				Gilgel Gibe I)		
4.	Gilgel Gibe III	1870 MW	Planned 2012-13	Omo River	Concrete gravity	420m x 610m
5.	Tekeze	300 MW	2009	Tekeze	Arch-gravity	185 m x 710 m
6.	Beles	460 MW	2010	Lake Tana (Blue Nile)	No	---
7.	Grand Renaissance Dam	6,000 MW	Planned 2014	Blue Nile River	Concrete Gravity	145 m x 1,800m

3.4 Dam Failures

The US based Federal Emergency Management Agency (FEMA) [30] defines dam failure as a "catastrophic type of failure characterized by the sudden, rapid, and uncontrolled release of impounded water or the likelihood of such uncontrolled release". Dam failure is considered to be something which causes unsatisfactory performance of the dam (or the portion thereof). Dam failures are comparatively rare, but can cause immense damage and loss of life when they occur. It results in a catastrophic break by a flood wave which results in a considerable loss of life or property.

Hundreds of dam failures have occurred throughout history. These failures have caused immense property and environmental damages and have taken thousands of lives. Table 3.2 [32] lists some of the recent dam failure incidences.

Table 3.2: Recent major dam failures

Dam name	Date	Country	Cause/reason
Campos de Goytacazes	January 4, 2012	Rio de Janeiro State, Brazil	Failed after a period of flooding.
Fujinuma Dam	March 11, 2011	Japan	Failed after 2011 Tōhoku earthquake.

Dam name	Date	Country	Cause/reason
Kenmare dam	October 8, 2010	Mozambique	Failure of tailings dam at titanium mine.
Hope Mills Dam	2010	North Carolina, United States	Sinkhole caused dam failure
Ka Loko Dam	2006	Kauai, Hawaii	Heavy rain and flooding. Several possible specific factors to include poor maintenance, lack of inspection.
Taum Sauk reservoir	2005	Lesterville, Missouri, United States	Computer/operator error; gauges intended to mark dam full were not respected; dam continued to fill. Minor leakages had also weakened the wall through piping.
Big Bay Dam	2004	Mississippi, United States	A small hole in the dam grew bigger and eventually led to failure.
Lawn Lake Dam	1982	Rocky Mountain National Park, United States	Outlet pipe erosion; dam under-maintained due to location
Wadi Qattara Dam	1979	Benghazi, Libya	Flooding beyond discharge and storage capacity damaged the main dam and destroyed the secondary dam in the scheme.
Sweetwater Dam	1916	San Diego County, California	Over-topped from flooding

Even though there is no recorded dam failure occurrence in Ethiopia so far, there are factors like sedimentation and possible occurrence of earthquakes which endanger dams and associated tunnels. The dam owner, Ethiopian Electric Power Corporation, has to put in place the concern of dam safety in order to extend the life span of dams and in order to exploit them up to their full potential by employing dam safety monitoring techniques.

3.5 Dam Safety Monitoring Techniques

The goals of safety monitoring of a dam is to identify, as soon as possible, any abnormality

which could be an indication of some upcoming danger in order to allow sufficient forewarning for the implementation of appropriate corrective measures. To accomplish these goals, regular assessments of dam condition and behavior as well as periodic safety evaluation are needed. Regular assessment serves to monitor the current health of the dam, whereas periodic safety evaluations serve to analyze long-term behavior in order to detect abnormal trends. Generally, the supervision of the structural behavior is ensured by applying the following procedure [21].

- Acquisition of monitoring information, i.e., instrument measurements as well as qualitative information from visual inspections;
- Interpretation and assessment of the information;
- Taking decisions;
- Archiving the results.

Currently available monitoring systems include:

Visual Inspection is a straightforward procedure that allows any properly trained person to make an accurate assessment of the dam's condition. The inspection involves careful examination of the surface on all parts of the structure. In the inspection process, the following equipment can be used: inspection checklist, notebook and pencil, tape recorder, camera which provides a reliable record of the observed field condition, hard level, probe, hard hat, pocket tape, flash light, rock hammer, binocular, gallon container and timer to measure the leakage flows [25].

Dam Instrumentation refers to the method and equipment used to make physical measurements of dams. However, instrumentation is not a substitute for inspection. It is a supplement to the visual observations made during an inspection. The requirements for instrumentation are more often specific to each individual dam site and require an evaluation by an engineer. Some of these instrumentations include Piezometers, Inclinometers, Tiltmeters, Crackmeters, and Jointmeters.

This instrumentation based monitoring methods, however, require cabling which is expensive. The cabling of a dam safety monitoring system is a complicated project, and the installation, adjustment and expansion of the monitoring points are also not sizeable as shown in Figure 3.4 [48].

Figure 3.4: Typical conventional structural health monitoring system

3.6 Summary

To summarize, this Chapter gives an introduction about dam safety and conventional dam safety monitoring procedures and instruments that are currently in use by dam owners which is so costly and complicated due to requirements of long cabling. Our alternative to dam safety monitoring system is based upon wireless networks composed of inexpensive distributed sensor nodes. The size, low cost, and expandability of wireless sensor nodes make them the ideal candidate for monitoring of dam safety. The model and design of the proposed solution is described in Chapter 5.

CHAPTER FOUR: RELATED WORK

In this Chapter, we summarize some of the related works that are published on the application of wireless sensor networks on Structural Health Monitoring (SHM) in general and dam safety in particular. Finally we conclude the chapter by showing the gaps that are intended to be filled by this thesis work.

Literature shows that there have been a number of works that have been done on dam safety monitoring. Here we have summarized those that are relevant and related to our study.

Zhu and Chai [9] made a survey on the use of wireless sensor networks in monitoring dam safety and presented a multi-agent architecture for collaborative wireless sensor network based dam safety monitoring. The architecture describes the multi-agent wireless sensor network based dam safety monitoring from hardware and software perspectives. At the network level, nodes connect together to form clustered topologies and nodes classified as sensor nodes which communicate wirelessly and connected to the sensor for dam safety monitoring, cluster heads which act as data aggregation and transmission, and station management node which is a more powerful desktop workstation that has wireless links and communicates with the cluster heads. From software architecture perspective, four types of software agents are identified. Structure monitoring agent is a software agent which accesses and makes available the sensor information of the represented sensor node to the dam safety monitoring application; Data manager agent is also a software which manages or mines regional sensor data. Application agent is the third software agent which communicates, cooperates and performs application tasks and the last software agent is User interface agent that allows to accept commands from the user and shows the result to the user. This work can be seen as groundwork on the adaption of multi-agent technology on dam safety monitoring using a wireless sensor network and it lacks detailed explanation and is not supported with experiment.

Xu *et al.* [11] suggested a mathematical technique called Dempster-Shafer theory as a method for dam safety evaluation in order to improve the accuracy of dam safety evaluation. It follows five steps to evaluate dam safety using Dempster-Shafer theory. First, it identifies the evaluation factors: horizontal displacement, foundation settlement, uplift pressure, cracking opening, and strain. Then, it determines the weighted value of evaluation factors impacting on the dam safety. Secondly, it establishes the member function of the evaluation factors. Thirdly, determines the

basic probability assignment based on Dempster-Shafer theory. Fourthly, synthesizes the total probability assignment based on Dempster-Shafer theory. Lastly, determines the level of security. The technique is shown to be highly effective with respect to similar techniques like average evidence theory and Fuzzy comprehensive assessment. This work can be used to evaluate dam safety by integrating it with a wireless sensor network.

Erfeng and Ziyang [12] tried to improve the effectiveness of a database system in the development of dam safety monitoring system and it analyzes the process and implementation of multi-dimensional database. But the database considered in this paper is not designed in consideration of wireless sensor based data collection and management for dam safety monitoring system.

Bettzieche [19] criticizes the existing computer-based monitoring solutions because they focus on remote monitoring and neglect a collaborative analysis of measured data. To address this, it proposes a monitoring system that incorporates a software agent that must conduct all of the tasks performed by experts involved in monitoring. The monitoring processes involve five steps. The first step is data acquisition stage, where the crew (team) supervised the dam according to monitoring plan. Secondly, checking plausibility which is just checking the measured data with respect to the previous measured data or on alarm values and are done by the measuring crew itself. Thirdly, checking of short team behavior by the responsible engineer by comparing with measurement of the previous week to find out anomalies in the short-time behavior of the dam and then transfer the data to the dam safety department of the company. Fourthly, checking long-time behavior of the dam by specialists at the head office of the company once in a month in order to find abnormal behavior in the long run. Finally, safety assessment is done once in a year by a responsible engineer on the overall data collected and supervises the measurements and analysis of data by using statistical tools and computer models. The main concern of this paper is to map the regularly performed tasks, the individual experts and the interaction between themselves onto a multi-agent system. This work can be taken as an innovative approach that is capable of demonstrating the enormous potential of agent-based application. However, the paper only focuses on the analysis of data gathered using a software agent which does not concern with the potential of wireless sensor networks on dam monitoring.

The work in [17] introduces the notion of water level monitoring and management within the

context of electrical conductivity of the water. The proposed methods comprise of a number of components which are LED light which indicate water level, self made water level sensor which senses the level of the water, PIC 16F84A micro-controller for controlling, monitoring, and water pump controlling system which controls the water pump. This water level monitoring and controlling also offers a method to control the system via the Internet. This monitoring system is suited for monitoring the water level of a tanker and is not scalable to monitor the water level of large dams due to its single sensor architecture and lack of communication with sensing devices.

Li *et al.* [18] designed a low cost water level monitoring system that is applied in Poyanghu Lake in China, which consists of a field sensor module which collects the water level and sends it to the base station module. The base station module comprises the hardware, e.g., micro-controller unit, serial expansion chip and software (ZKOS-home made operating system), a data center module and web realizing module. The paper clams that it has the advantage over the current related systems in providing real-time and synchronized remote control, expandability, and anti-jamming capabilities. The paper also outlines its future works and its recommendation as follows: combination of more nodes, intelligent power management, and integration with remote sensing systems. The paper mainly emphasizes on the on water level monitoring which is a fraction of parameters that need to be deployed in dam safety monitoring.

Xinying *et al.* [36] developed and simulated a wireless sensor network for dam monitoring, WDSN (Wireless Dam Sensor Network). Their system consists of sensor nodes with cluster head, sink nodes, and a computer management center. It is based on network cluster architecture that incorporates a multi-hop and clustering with the aim of lowering energy consumption. They also model the energy loss due to propagation distance using free space propagation for a distance between the cluster heads and the sensor nodes and two path model for a distance between the sink nodes and the cluster nodes. Simulation test was performed using random node deployment strategies in the detection region with area of 100m x 100m locating sink node far from the closest sensor node at x=50, y=160 and the number of sensor nodes range from 10 to 100. They have compared their system with typical WSN with multi-hop structures and the clustering structure on the base of the LEACH protocol energy consumption mode. They concluded that the result shows it is reliable transmission, and also shows the maximum transmission distance. Maximum transmission distance is achieved when the distance of adjacent

nodes is increased to 300m. Thus, the demonstrated WDSN has the potential to a wide range of dam monitoring. However, they do not consider the requirements of a different parameter to be monitored in dam safety.

A public alert system for dam discharge using a wireless sensor network is presented in [52], which applies the wireless sensor network technology to monitoring and alerting flooding incident in association with dam discharge. Using a wireless sensor network, adaptive query, and cell broadcasting service, the public alert system for dam discharge provides the right alert information to the right person at the right time. As proof of this concept, they have developed a prototype dam discharge alert system that employs the adaptive querying technique on top of a wireless sensor network to implement real-time alert service. Thus, their system minimizes flood damage related to dam discharge through prediction, dynamic data collection, and an alert service in emergency conditions.

The work in [53] discusses the design and evaluation of a wireless sensor network for structural data acquisition called Wisden. A Wisden deployment typically comprises tens of wireless nodes (placed at various locations on a large structure) that self-configure to form a tree topology and reliably send time-synchronized vibration data to the sink (the tree root), over multiple hops. The sink forwards this data to a base station, usually a high-end PC. They implement these mechanisms on the Mica2 nodes that measure structural vibrations with the help of a vibration card specifically designed for the high-quality, low-power vibration sensing suitable for structural health monitoring applications and evaluate the performance of the implementation. Wisden incorporates reliable data transport using a hybrid of end-to-end and hop-by-hop recovery, and low-overhead data time-stamping that does not require global clock synchronization. They also studied the applicability of wavelet-based compression techniques to overcome the bandwidth limitations imposed by low power wireless radios. As a proof of this application, they deployed Wisden in real environments and the experiments proved that Wisden required minimum amount of time for set up which took several days to set up wired data acquisition system. This demonstrates the fundamental rationale behind wireless sensor networks: flexibility and ease of deployment.

The work in [55] investigates a wireless sensor network deployment for monitoring water quality, (e.g. salinity and the level of the underground water table), in a remote tropical area of

northern Australia. Its goal is to collect real time water quality measurements together with the amount of water being pumped out in the area, and investigate the impacts of current irrigation practice on the environment, in particular underground water salination. They have designed, implemented and deployed a sensor network system, which has been collecting water quality and flow measurements, e.g., water flow rate and water flow ticks for over one month and the paper concludes that the preliminary results show that sensor networks are a promising solution to deploying a sustainable irrigation system, e.g., maximizing the amount of water pumped out from an area with minimum impact on water quality.

The work in [57] proposes a cyber-physical co-design approach to structural health monitoring based on wireless sensor networks. The approach is different from the existing research approach where the design of wireless sensor networks and structural engineering algorithms are separated. The paper aims to integrate flexibility based damage location method and an energy-efficient, multi level computing architecture. The intuition behind flexibility based methods is that structures will bend slightly when a force is applied and as a structure weakens, its stiffness decreases, and thus its flexibility changes. Changes in structural flexibility over a structure's lifetime can be used to identify and localize damage. The proposed system works as follows: In the first stage of the multi-level search, minimal numbers of sensors are enabled, forming a single cluster. In the event that damage is identified, the flexibility-based method will also output coarse-grained damage localization. This second round subsequently localizes the damage to a smaller region than the first round. The system may repeat this drill-down procedure to achieve even finer grained results until the desired resolution is reached. The key feature of this approach is that it does not activate the entire sensor network at once. Instead, relatively few sensors are used to identify damage; and when damage is identified, only those sensors in the area of interest are incrementally added to the search. Once the nodes participating in this multi-level search are selected, they are each assigned one of three different roles: cluster member, cluster head, and base station. A node's role determines what data it handles as well as its level in the network hierarchy. To allow the system to better scale to large structures, the nodes may be organized into multiple independent clusters. Each cluster operates as an independent unit, with the cluster head coordinating nodes within its cluster and ultimately transmitting the cluster's mode shape data to the base station for signal processing. Based on these roles, the system operates as follows. The cluster members collect raw vibration samples from their on-board accelerometers.

They then carry out the Fast Fourier Transform (FFT) to transform the vibration response into frequency domain data, followed by a power spectrum analysis. They have implemented their system on top of the Imote2 sensor platform using the TinyOS operating system and utilize the Illinois Structural Health Monitoring (ISHM) services toolsuite developed by the Illinois Structural Health Monitoring Project (ISHMP), which provides subsystems for sensor data acquisition, reliable data transmission, and time synchronization based on the TDMA transmission schedule. Their experimental results show that their system is able to localize damage to the resolution of a single element on a representative physical beam and simulated structures. They also demonstrated the energy efficiency of this approach through latency and energy consumption measurements.

The work in [20] describes a WSN platform for detection of structural state through the measurement and interpretation of ambient vibrations and strong motion choosing Golden Gate Bridge in San Francisco as a test bed. The network consists of 64 nodes and a base station, which measure ambient vibrations. The software architecture of the Golden Gate Bridge nodes uses components integrated into the TinyOS infrastructure to satisfy the requirements. The goal is to have reliable and lossless communication over a large network with minimal overhead for other network components. To this end, they designed and implemented the Straw (Scalable Thin and Rapid Amassment Without loss) component, a reliable, highly scalable, data collection service. Straw works over a multi-hop routing layer, with transfer initiated by the receiver. Since it is a collection protocol, the receiver is always a PC, and the sender a node. First, requirements are identified to obtain data of sufficient quality to have real scientific value to civil engineering researchers for structural health monitoring. An accurate data acquisition system, high-frequency sampling with low jitter and time synchronized sampling are provided in this work which are crucial for data to be useful for structural health monitoring. Second, the system is designed to scale to a large number of nodes to allow dense sensor coverage of real world structures. This is verified on a 64-node, 46-hop deployment over the main span and a tower of the Golden Gate Bridge. Third, this network is deployed in a real world structure solving a myriad of problems encountered in a real deployment in difficult conditions. The researcher in this paper concluded that the network provided reliable and calibrated data for analysis, which was not possible in previous studies.

Land slide detection using wireless sensor network is presented in [54], which describes the implementation of a wireless sensor network system for landslide detection. It mainly focuses on the design, development and deployment of a wireless sensor network, the development of data collection and data aggregation algorithms needed for the network, and data analysis system. It is a heterogeneous network composed of MicaZ and STARGATE wireless sensor nodes, Wi-Fi, and satellite terminals for efficient delivery of real time data to the data management center. The system also has data management center which is equipped with software and hardware needed for sophisticated analysis of the data. The results of the analysis in the form of landslide warnings and risk assessments will be provided to the inhabitants of the region.

Our solution is intended to fulfill the following issues which were not addressed by the works reviewed.

1. Consider wireless sensor nodes which sense multiple parameters that are likely to be used for dam safety like displacement, seepage, uplift pressure, etc.
2. Model a distributed application that incorporates data collection, routing, data aggregation, data management, and data visualization services for dam safety monitoring system using wireless sensor networks.

CHAPTER FIVE: PROPOSED SOLUTION

5.1 Introduction

As described in Chapter 3, the current conventional dam safety monitoring techniques are so costly and complicated due to requirement of long cabling and expensive devices. In this thesis, we proposed a dam safety monitoring application using wireless sensor network utilizing small, smart, inexpensive nodes for real-time and distributed data acquisition, routing, data management, and visualization to achieve the goals of dam safety monitoring and in turn prevent failure, reduce maintenance costs, and extend the lifetime of dam.

Following this introduction, in the following sections we have described the details about the techniques and the model developed for the proposed solution. In Section 5.2, we describe and present the proposed system architecture for dam safety monitoring using wireless sensor networks. Section 5.3 summarizes the sensor nodes requirements and assumptions for dam safety monitoring system purpose. These requirements set the boundary for the dam safety monitoring application and routing protocol presented in Section 5.4. Finally, Section 5.5 summarizes this chapter.

5.2 Proposed System Architecture

In this section, we describe and present the proposed system architecture for dam safety monitoring using wireless sensor network as illustrated in Figure 5.1. The architecture shows the heterogynous sensor nodes that are organized in multi-hop cluster based tree to sample the different physical parameters of the dam and route the reading to a central place for data management and visualization purpose.

The architecture accommodates heterogynous sensors to sample the different physical parameters of the dam, for example, some of the nodes monitor the water level, some measure uplift pressure, and some monitor seepage while others measure settlement. The architecture requires the sensor nodes to organize in multi-hop cluster based tree networks in order to allow sensor nodes deliver reliably the data between nodes, due to the requirements of large number of sensor nodes installed on the dam surface, some of which are out of direct communication range with the base station node and also the architecture enables clustering sensor nodes to aggregate

the sample at the cluster head which reduce battery usage by reducing the amount of transmission to the base station node.

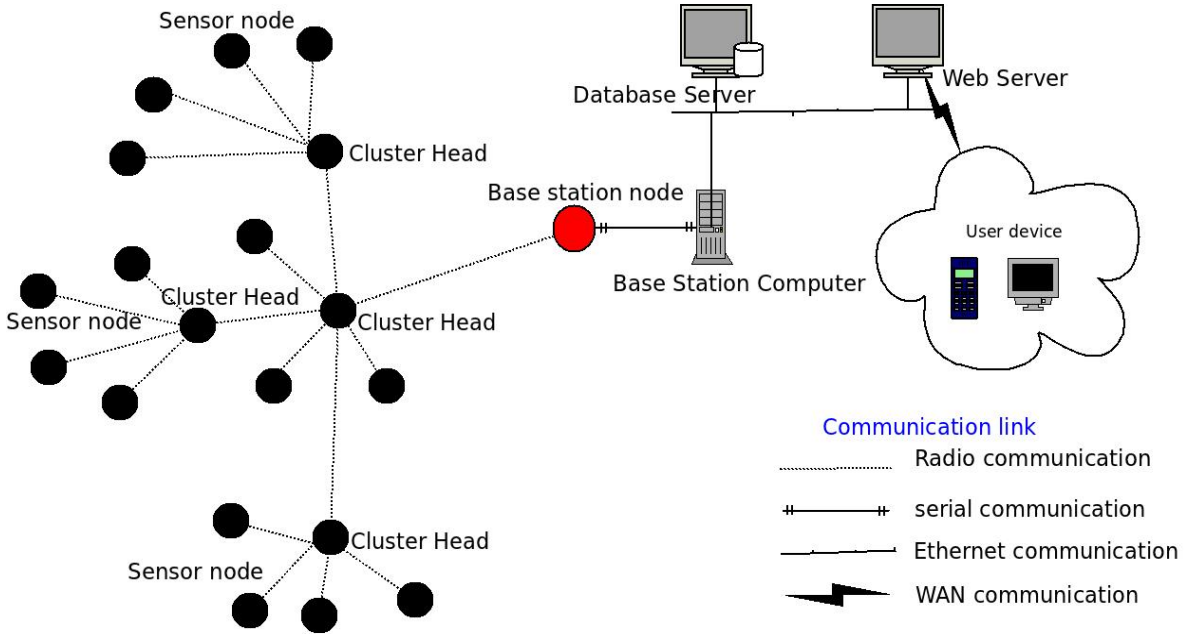


Figure 5.1: Architecture of the proposed solution

The communication methods are also indicated in the architecture where the transfer of packets between sensor nodes and cluster head, cluster head and base station is made through radio communication. The base station node communicates with base station computer using serial communication to send the samples from the sensor networks to the data management center. The base station computer then forwards to the database server via Ethernet communication. The dam safety monitoring personnel get the status of the dam through web interface using Internet or SMS notification using GSM network.

5.3 Sensor Node Hardware

Motivated by the vision of having low cost and small size smart sensor nodes which can collect sample about the condition of the dam, we first consider the hardware platform requirements for our application. We divide the hardware components into (1) sensing, (2) memory, (3) communication, (4) processing, and (5) energy supply. This section summarizes the requirements in each of the above mentioned components and consequently makes an argument for choosing a specific hardware platform.

Regarding sensing, we choose the vibration type sensors for our prototype since we are primarily interested with recording of vibration response due to water pressure (directly related with the water level of the dam), water flow through seepage, change in settlement of the dam, and uplift pressure on the dam bottom surface. However, the selection of sensor types used for dam monitoring is determined by type and size of dam. In this thesis we select sensors which are commonly used in dam safety monitoring and we demonstrate the task of these sensors operation using "demo sensor[1]" which simulate the response of vibration sensor to determine the different measurements parameter of the dam.

The memory requirements arise from buffering the sample from the sensor, holding intermediate result of certain computation, and also storing packet that forwarded from other sensor nodes to be routed to the base station. For communication, we choose for wireless support since in general, different sensor nodes which physically separated at a distance should be able to collaboratively perform data collection. The processing requirements are minimal, only simple mathematical operation and data aggregation. Finally, the device must be powered by a battery that last long.

In regard with the above requirements, we consider MicaZ sensor nodes which have an 8-bit microcontroller, use an IEEE 802.15.4 compliant radio, use two AA batteries, and also include a 512KB memory to prototype our application.

5.4 Dam Safety Monitoring

Each sensor node runs monitoring application and routing protocol to read and transmit packets from the sensor node to the cluster node then to the base station. The base station computer also runs an application to receive sampled data and stores in the database. We have shown the chain of software tools in Figure 5.2 that are designed in this thesis to achieve the goal of the dam safety monitoring.

[1] Demo sensor is a program that simulates the task of real sensor.

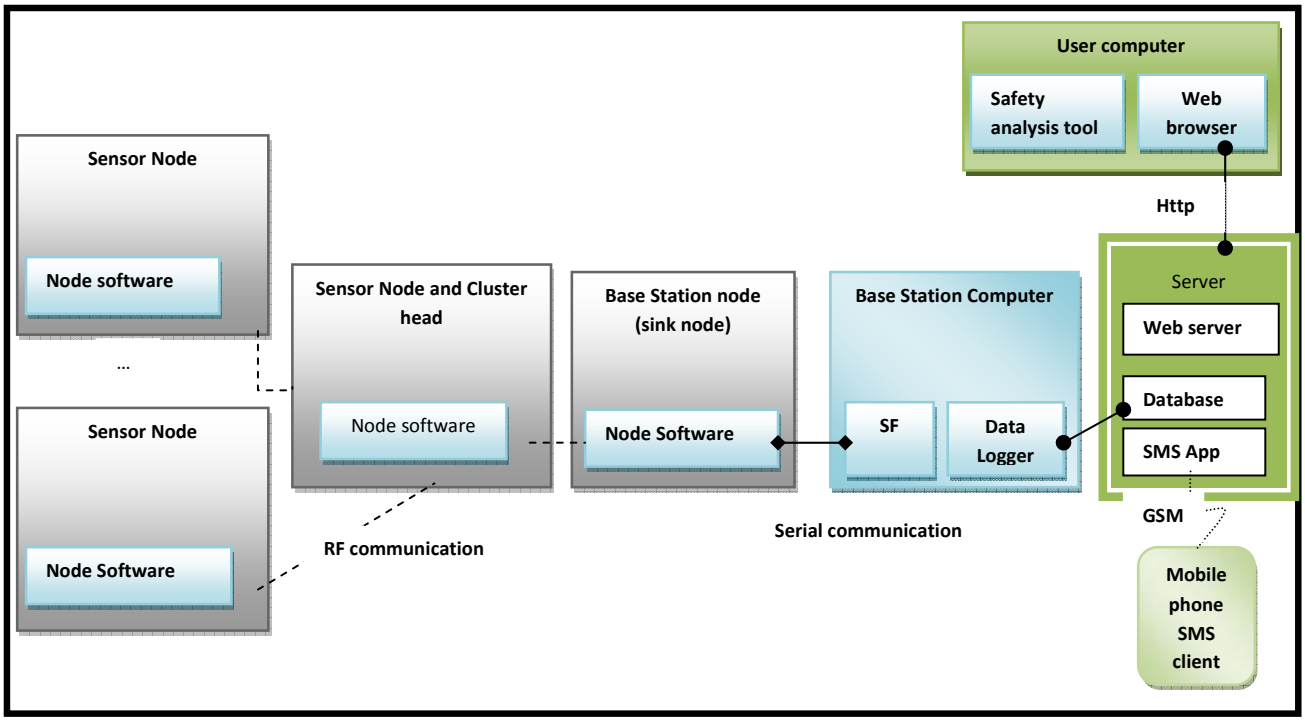


Figure 5.2: Chain of Software tools (wireless sensor network application, data management, and visualization)

As shown in Figure 5.2, data collection begins when the sensor nodes read their sample and send to their cluster heads and finally to the base station node through wireless communication. The base station node then transfers the sample reading to the base station computer through its UART port in serial communication as serial packet. The base station computer then receives and retrieves the serial packet using SF tool. The SF tool forwards the packet to the data logger application which runs on the base station computer. Upon receiving a packet from SF, data logger application extracts the sensor reading value from the serial packet and then stores the value in the database. The base station computer and the database server communicate through TCP/IP protocols. The web server reads the database records and visualize to the user through web interface. SMS application sends alarm notification to dam safety personnel for an immediate action to be taken by accessing the database and performing trend analysis.

The design details of these software tools that are used by the sensor nodes, base station computer, database server, and web server are described in the following sub section.

5.4.1 Dam Safety Monitoring (DamSafeMon) Application

DamSafeMon is our proposed wireless sensor network software which we design for dam safety monitoring application data sampling, routing, and tasks. The sensor nodes run data acquisition application that determines when and how to sample monitoring data using their sensor. Figure 5.3 shows the DamSafeMon stack on a node; the node's upper layer provides an interface and builds on our application specific routing protocol and adaptive data sampling services.

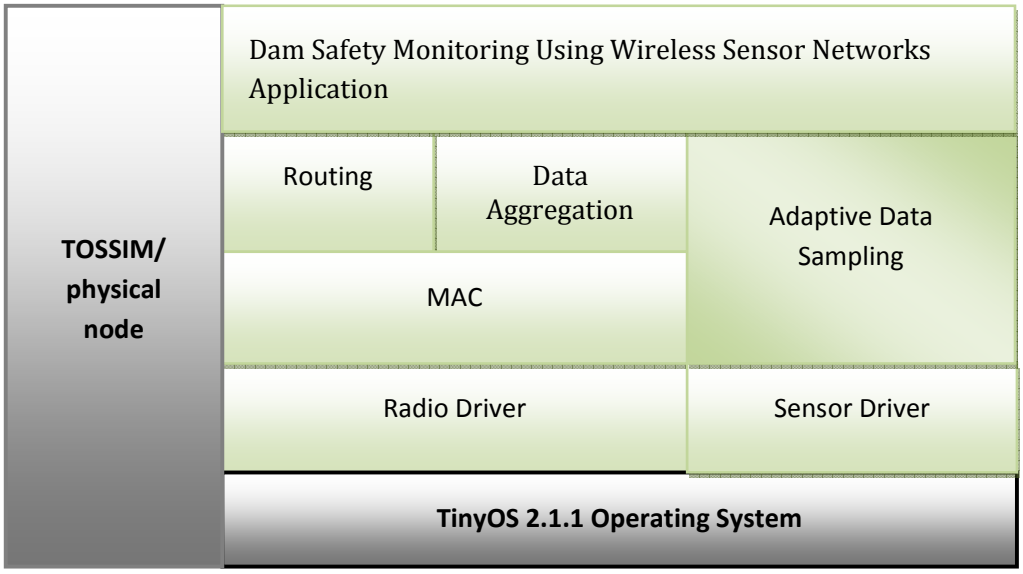


Figure 5.3: Software stuck of DamSafeMon on Sensor Node

Adaptive Data Sampling: We proposed a simple adaptive data sampling technique, as shown in Figure 5.3, to determine when the sensor nodes should make sampling about the physical parameter of the dam in contrast with the existing periodic sampling method which makes sensor nodes to take reading sample data regularly.

In our sampling techniques the sensor nodes adjust their next sampling time based on the previous sampled data. Meaning that, if the difference of two successive sensor readings is greater than the threshold value (tolerable value between two successive readings), then increase the sampling frequency by 2. This reduces the energy consumption of the sensor node by avoiding sampling similar reading sample if the consecutive readings are the same.

Let the default sampling frequency for a given sensor type be, $default\_samp\_period$, current reading $S(t_i)$ and previous reading $S(t_{i-1})$. The next sampling time is determined by the equation below.

$$samp\_period = \begin{cases} samp\_period / 2 & if\ /S(t_i) - S(t_{i-1})/ > threshold \\ default\_samp\_period & if\ /S(t_i) - S(t_{i-1})/ \leq threshod \end{cases}$$

Thus the sensor nodes are programmed to take the next sample which is regulated by adaptive techniques shown in the formula above. The algorithm for adaptive sampling and data acquisition application is shown in Figure 5.4.

Input: *default_samp_period and threshold*

Process

1. *Samp_period ← default_samp_period*
2. *IF (sensing_timer_expire)*
3. *reading ←data from ADC(sensing)*
4. *Call Routing protocol to forward the reading*
5. *ENDIF*
6. *IF(current reading - previous reading >threshold)*
7. *samp_period ← samp_period/2*
8. *Next_sampling_time ← current_time+samp_period*
9. *ELSE*
10. *Next_sampling_time ←current_time+default_samp_period*
11.
12. *ENDIF*

Output: *Current sampled data and next sampling time*

Figure 5.4: Adaptive sampling

Multi-Hop Cluster based Routing Protocols (MHCRP): Generic multi-hop routing protocols allow sensors to cooperate to reliably deliver data between nodes outside of direct communication range. However, application specific requirements, such as node deployment strategies, data reporting model, energy constraint, are quite challenging to achieve with generic multi-hop communication protocols [38].

Thus, we designed an application specific Multi-Hop Cluster based Routing Protocols (MHCRP) that allows a reliable transfer of sampled data from the sensor node to the base station that meets the requirements of the architecture described in Section 5.2. It is a cluster based routing protocol in which self-elected cluster heads collect data from all the sensor nodes in their cluster, aggregate the collected data by data aggregation methods and transmit the data through an optimal path between the cluster head (CH) and the base station (BS) through other intermediate CHs and use these CHs as relay stations to transmit data through them as shown in Figure 5.1.

In our model sensor nodes are grouped into different clusters. Each cluster is composed of one cluster head (CH) and cluster member nodes. The respective CH gets the sensed data from its cluster member nodes, aggregates the sensed information and then sends it to the Base Station through an optimal multi hop cluster based tree formed between cluster heads (CHs) and base station. In our application, the sensor nodes are deployed in region of large dam; it may be beneficial to use multi-hop communication among the nodes in the cluster to reach the cluster head.

Our clustering process is based on LEACH [42] clustering technique. However, we modified it to meet the requirement of our application that there must be at least one cluster head for each sensor category in each round. Sensor nodes are categorized based on the type of data they sample. For example, nodes that sample water level are grouped in a single category. Our clustering process, as LEACH clustering technique, performs in two phases: the setup phase and the steady state data transfer phase. In the set up phase, the clusters are organized and cluster heads selected based on the suggested percentage cluster head in the network and the number of times the node has been a cluster-head so far in each sensor category. This decision is made by each node *n* choosing a random number between 0 and 1. If the number is less than a threshold *T (n)*, the node becomes a cluster-head for the current round. The threshold *T (n)* is set as follows:

$$T(n) = \begin{cases} \dfrac{p}{1 - p * (r \bmod \frac{1}{p})} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases}$$

where p is the desired cluster-head percentage in each sensor category which is given by the user, r is the current round and G is the set of nodes in each sensor category that have not been cluster-heads in the last $1/p$ rounds. The process of clustering task of the modified LEACH protocol is demonstrated by the pseudo code shown in Figure 5.5.

Input: *number of cluster head node needed in the network*

Process

1. *Node choose random number(rand) between 0 and 1*
2. *Computer T(n) as formula shown above*
3. *IF (rand < T(n))*
4. *Node(n) become cluster head for current round*
5. *ELSE*
6. *Node(n)) not cluster head for current round*
7. *ENDIF*

Output: *cluster head nodes*

Figure 5.5: Clustering process

Once the sensor nodes have elected themselves to be cluster heads, they broadcast an advertisement message to make other nodes update their routing table. A non cluster head node then joins the cluster head node selecting from the lists in its routing table that fulfill the criteria that the cluster head have good link quality and identical sensor type with the non cluster node. The sensor nodes determine the quality of the communication link between other nodes using expected transmission (ETX) techniques as described in Chapter 3. Figure 5.6 show the pseudo code of cluster head selection process by the non cluster head node.

Input: *routing table, link quality*

Process

1. *IF the Node(n) is sink node*
2. *Return*
3. *Else*
4. *For each node in Routing Table*
5. *IF (node is a cluster head and minimum ETX and similar sensor type*
6. *Join the cluster head*
7. *ENDIF*
8. *ENDFOR*
9. *ENDIF*

Output: *cluster head selected*

Figure 5.6: Pseudo-code of cluster head selection process by non-cluster node

A cluster node again joins another cluster head node to make a multi hop cluster tree. In such case the cluster node should select a cluster node that has a good link quality towards the sink node. Figure 5.7 show the pseudo code of parent selection process by the cluster head to make multi-hop cluster tree.

Input: *routing table, link quality*

Process

1. *IF Node(n) is sink node*
2. *Return*
3. *Else*
4. *For each node in Routing Table*
5. *IF node is a cluster head and minimum ETX with sink node*
6. *Join the parent and update its link quality*
7. *ENDFOR*
8. *ENDIF*

Output: *parent selected*

Figure 5.7: Pseudo-code of parent selection process by cluster head

In the steady state phase, the actual data transfer to the base station takes place. Upon receiving all the data from its cluster member nodes, the cluster head node aggregates it before sending it to the other cluster head nodes.

5.4.2 Data Management, Safety Analysis, and Visualization

In our dam safety monitoring application, presenting the data to the user in a user-friendly manner and providing exact detail is one of the requirements. A natural solution to address the requirement is to log each sample data that come from the sensor networks into database and design safety analysis and visualization method on top of the database. Figure 5.8 illustrates the design of data logger application which forward data from sensor network and store in database, the logical design of the sensor database, and finally the design of web application and SMS notification module.

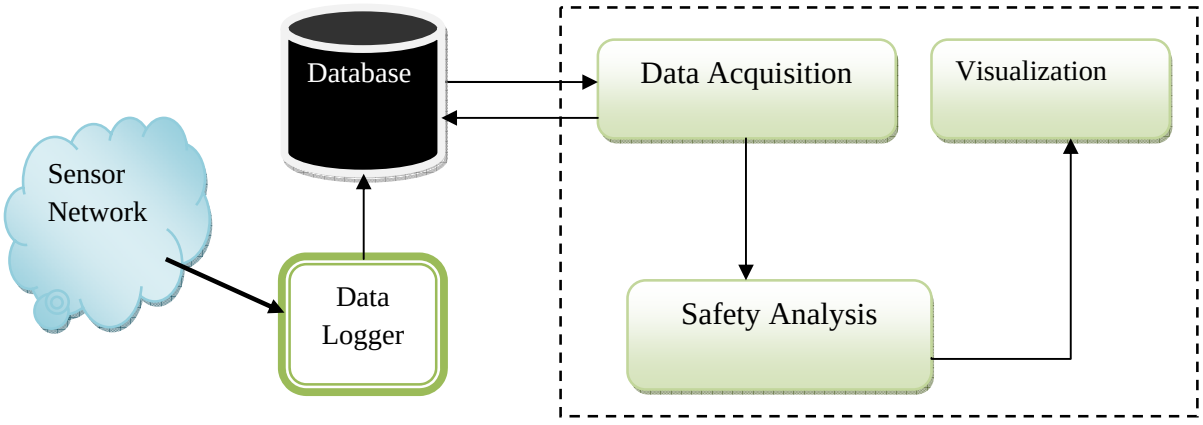


Figure 5.8: Data management, Safety Analysis, and Visualization sub-system

Data Logger Application: Packet data which we get from the sensor networks includes not only the sensor reading but also other information like packet source address. The sensor reading (sample) must be separated from the other packet detail in order to be stored in the database. Thus, the data logger application extract sensor sampled data from the serial packet and store it in the database. Figure 5.9 shows the operation of the data logger application described in the pseudo code.

Input: *packet from serial port of the base station*

Process:

1. *Get the size of the packet*
2. *IF (Size ==standard packet Size of Serial packet)*
3. *Extract reading data from packet*
4. *Store in the database*
5. *Else (Size ==standard packet Size of Serial packet)*
6. *Invalid packet*
7. *ENDIF*

Output: *Display the value on GUI and at the same time store it in database*

Figure 5.9: Pseudo- code of data logger application

Database System: The information gathered by the sensor nodes has to be stored in data storage system which used for dam safety analysis and evaluation. The information kept by the database consists of information about the deployed sensor nodes, the value of parameter they read, the geographical information of each node. The data obtained from sensor network is saved as specified data format in the designed database and the database allows making query about the condition of the dam.

Keeping this in mind, we design the monitoring database using relational database model. Two main entities are identified after analyzing our dam safety monitoring requirement. The first entity, NODE_INFO, that holds information about the sensor nodes like node identification number and type of parameter it measures whereas the second entities, MEASUREMENT, which is a generalized entity that encapsulate WATER_LEVEL_MEASUREMENT, UPLIFT_MEASUREMENT, SEEPAGE_MEASUREMENT, and SETTLEMENT_MEASUREMENT that holds about the reading value by the sensor and the time of measurement.

Figure 5.10 shows the design of the relational database for dam safety monitoring application but detail design and its implementation described in Chapter 6.

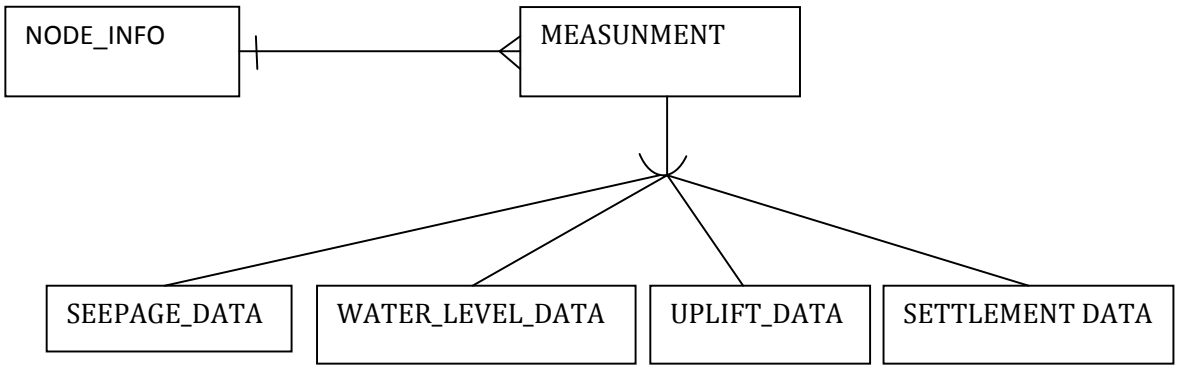


Figure 5.10: Database design for dam safety monitoring

Web Interface: The dam safety monitoring data can be presented to the user through Internet using web application. The web application designed to visualize:

- Monitoring parameter trend chart: The monitoring parameter trend chart of the designed dam can be drawn in our dam safety monitoring application through the time period.
- Event list: Special event that are unusual are logged in and visualized to the user to trace the history of such event.
- Dam map and sensor location: The dam map along the position of the sensor in the map is introduced in our dam safety monitoring application to show the location of sensor nodes, the status (working or failed) of the node.

SMS Notification: The dam monitoring alarm notification can be obtained by SMS and it is very useful to notify the dam safety operator when unusual event occurred. Alarm notification application pseudo code described and shown in the Figure 5.11.

Input: *threshold value, trend time frame*
Process:

1. *IF (trend Time Frame expire)*
2. *Query database to analyze the trend (for a given time frame)*
3. *IF (value of the trend > threshold value)*
4. *Alarm message*
5. *ENDIF*
6. *ENDIF*

Output: *SMS message*

Figure 5.11: Pseudo-code for alarm notification

5.5 Summary

In this Chapter, we have described the proposed architecture for dam safety monitoring application and also the sensor node requirements for dam safety monitoring. Following this we also describe the design of sensor node application and routing protocol. Lastly, we describe the design of data management and visualization tools. In the next chapter we will describe the implementation detail of the proposed safety monitoring application and also the multi-hop cluster based routing protocol, and then data management and visualization tools.

CHAPTER SIX: PROTOTYPE IMPLEMENTATION AND EVALUATION

6.1 Introduction

In the monitoring process, the wireless sensor nodes sense the physical status of the dam, then send its reading to its cluster head node and finally reached to sink node through wireless channel. The sink node which is connected to the base station computer through a serial interface; act as a proxy between the wireless sensor network and the data management unit. The base station computer then forwards the received information from the sensor node to a database server, where the data is stored, analyzed, and finally visualized to dam monitoring personnel via Internet and SMS.

To accomplish the above tasks, we implement the following tools: wireless sensor network application, multi-hop cluster based routing protocol, data management, and visualization tools. The implementation detail description of these tools is presented in various Sections of this Chapter. Section 6.2 gives an overview of prototype description. Section 6.3 describes the development environment employed to implement the system. Section 6.4 presents the implementation description of the various components. In Section 6.5, the simulation experiment and evaluation result is described. Finally, we summarize the Chapter in Section 6.6.

6.2 Development and Simulation Tools

The development environment, programming language and simulation tool that were used for implementation and evaluation of the proposed solution is described in this Section.

We have used TinyOS 2.2.1, a defacto operating system for wireless sensor nodes; since it becomes the preferred choice of the research community to design and implement wireless sensor network systems. It is free and open source software and can be download from [2]. It is written using nesC [3] programming language and we also used this programming language to write the monitoring application.

TOSSIM-V2 [4] simulator is used which is the default TinyOS simulation tool, which simulates the entire application by replacing the missing functionality of hardware components by simulation implementation. However, TOSSIM currently supports only MicaZ sensor node so that the application has to be adapted and compiled specifically for this kind of sensor. The current version of TOSSIM does not capture the power consumption of an application running

on a given node. Thus, we used PowerTOSSIMZ [49], which is a dedicated plug-in for TOSSIM simulator, to model the power consumption of MicaZ node.

Serial Forwarder (SF) is another tool that is used to read the packets from sink sensor node serial port to base station computer in order to allow transfer of data between the sensor network and data management unit. The packets which are reached at base station computer are captured and forwarded to data logger application which in turn extracts the packet payload and then stores it in the database. The database designed and implemented is using MySQL database management system tool. The SMS notification tool is implemented using python programming language, MySQLdb [34], and python Serial [35] library. PHP and JpGraph [44], graph plotting library, are used to implement the web application.

6.3 Prototype Implementation

In the previous section, the necessary tools are identified for designing and implementing of the dam safety monitoring application. This section describes the implementation detail of the different components of the monitoring application. Section 6.4.1 describes the implementation of wireless sensor network application which runs on the sensor nodes. Section 6.4.2 discusses the implementation of application specific routing protocol which is designed specifically for our monitoring application. Section 6.4.3 describes the implementation detail of the data logger application. In Section 6.4.4, the database design and implementation is described. Section 6.4.5 describes the implementation of the web application. Finally, Section 6.4.5 describes the implementation of SMS notification module.

6.3.1 DamSafeMon: A Dam Safety Monitoring Application

DamSafeMon is an application that runs on all the sensor nodes in the network to monitor the safety of the dam which is implemented using nesC programming language on TinyOS operating system. In the implementation process of this application, the DamSafeMon is represented as components:

- *SensorNodeC* – A component which defines the task of monitoring application.
- *DemoSensorC* – A component which defines the driver of the sensor
- *mhcrpC* – A component which defines the task of multi-hop routing.

In addition to the components defined above, DamSafeMon utilizes components that are defined and implemented in TinyOS operating system.

- *MainC* – A component that defines the task of booting the sensor node.
- *TimerMilliC* – A component which defines the virtualized millisecond timer abstraction.
- *ActiveMessage* – A generic component that defines the task of sending radio packet.
- *SerialActiveMessage* – A generic component that defines the task of sending packet over the serial port.

These components are then wired with each other using interface to accomplish the tasks of DamSafeMon application as shown in Figure 6.1.

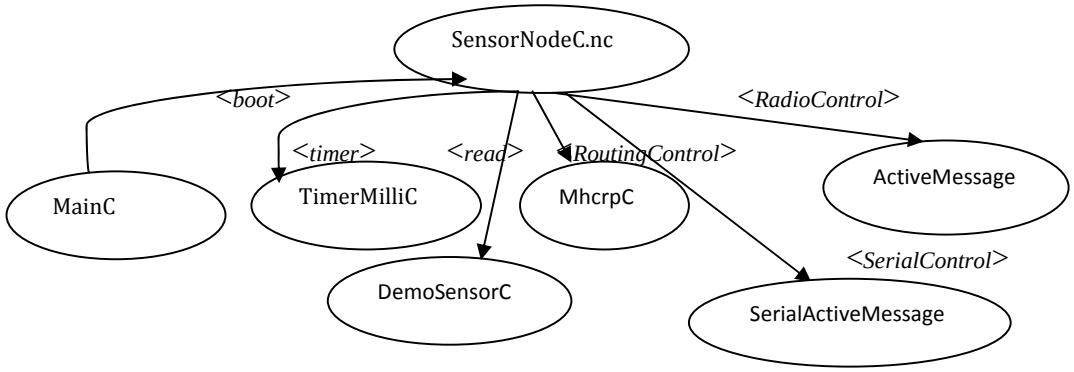


Figure 6.1: Components and interface graph

When a boot event occurs, *SensorNodeC* turns on the radio of the sensor node using *RadioControl* interface which wire the application with radio of sensor node. Once the radio is on, the application starts the routing sub-system using *RoutingControl* interface.

When a radio packet arrives to the sink node from other node, i.e., when *RadioReceive* event occurs, the application determines the type of the received packet where the packet can be sampled as data packet or control packet. Once the sampled data packet reaches to the sink node, it converts the received radio packet to the serial packet format before sending it to the base station computer through serial port. The application uses a component, *SerialActiveMessage*, for sending serial packet using an interface *SerialControl*.

The application samples data by ordering *DemoSensor.nc* component with *Read* Interface. Though *DemoSensor* is a generic component, we have defined sensor specific components which are used by our dam safety monitoring application: *VibrationWaterLevelSensorC* is a component which measures water level, *VibrationUpliftSensorC* is a component which measures uplift pressure, *VibrationSeepageSensorC* measures seepage water flow, and *VibrationSettlementSensorSensorC* measures the settlement change.

We have employed application level timers. A *Timer0*, an instance of component *TimerMilliC*, is used to determine when to make data sample by the sensor. As we have modeled in Chapter 5, we used adaptive sampling, where the application doubles the rate of sampling when the reading by the sensor increases from the previous reading above the threshold. A *Timer1* is again another instance of component *TimerMilliC* which is used for determining when to on/off the radio module.

The nesC source code for the modules *SensorNodeC.nc* and other configuration file is found in appendix A.

6.3.2 MHCRP: A Multi-hop Cluster based Routing Protocol

MHCRP is the application specific cluster based routing protocol. This section describes the implementation of MHCRP routing protocol that has been proposed and modeled in Chapter 5. MHCRP extends the existing tree based routing protocol, CTP, which was implemented in TinyOS 2.1.1 by adding: a multi-hop cluster tree formation logic that is executed prior to parent selection, and a packet aggregation mechanism in the cluster head node. Similar with the CTP routing protocol, our routing protocol is composed of three modules namely Routing Engine module, Link Estimator module and Forwarding Engine module as shown in Figure 6.2. MHCRP modifies the Routing Engine module to implement a multi-hop cluster based tree construction and we also modified the Forwarding Engine module implements an aggregation operation on the cluster head node. We did not need to modify the CTP Link Estimation module at all and we used as it is.

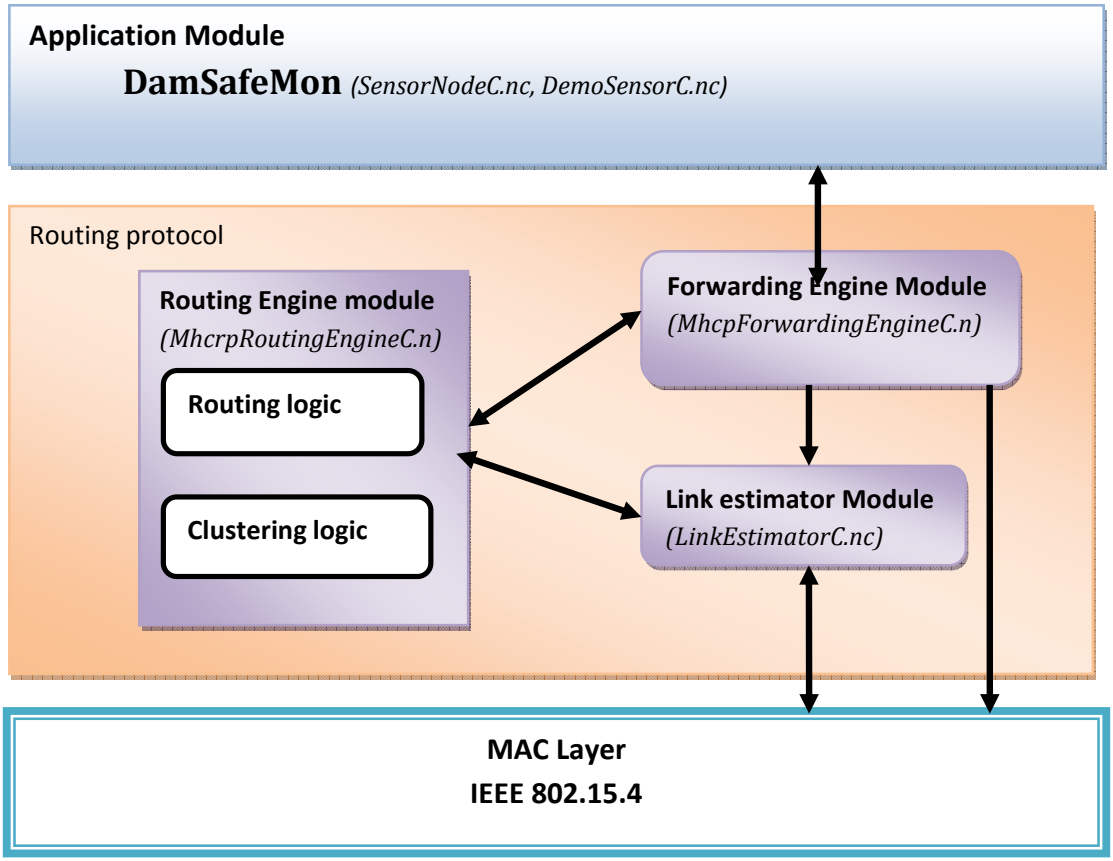
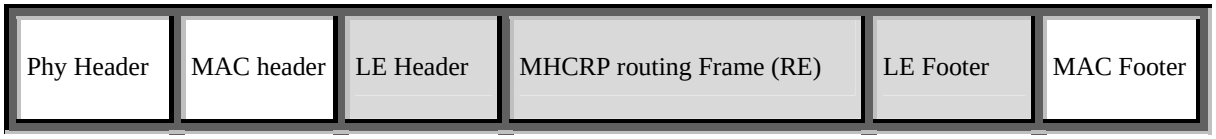


Figure 6.2: Schematic design of the MHCRP routing protocol

Routing Engine Module: In our implementation, the main tasks of the Routing Engine consist of sending beacons, filling the routing table and keeping it up to date, electing and joining cluster head node, and selecting parent node in the multi-hop cluster based routing tree towards which data frames must be routed. For such purpose we define a module, *MhcrpRoutingEngineC*, which performs the above mentioned tasks and a routing frame which is 9 bytes long as shown in Figure 6.3 (a), which contains information that help for a multi-hop cluster tree construction.

0		8	15
P	C	Reserved	Parent
Parent			Cluster head
Cluster head			ETX
ETX			Sensor type
CollectId			

(a) Routing frame



(b) Routing packet

Figure 6.3: Structure of MHCRP's (a) routing frame and (b) routing packet

The routing frame is 9 bytes: The first byte of the MHCRP routing frame defines the P(Pull) and C(Congestion) flags. The former is used to trigger the sending of beacon frames from neighbors for topology update, while the latter allows a node to signal that it is congested. If a node receives a routing beacon from a neighbor with the C flag set, it will stop sending packets to this neighbor and look for alternatives routes. The last 6 bits of the first byte are reserved for future use. The second and third bytes define the parent field which specifies the identifier of the parent of the sender which sends the beacon. The fourth and fifth bytes define the cluster head field which specifies the identifier of the cluster head. The sixth and seventh bytes include the multi-hop ETX metric. The eighth byte specifies the category of the sensor type. Finally, the ninth byte specifies the identification of the network if more than one sensor network exists.

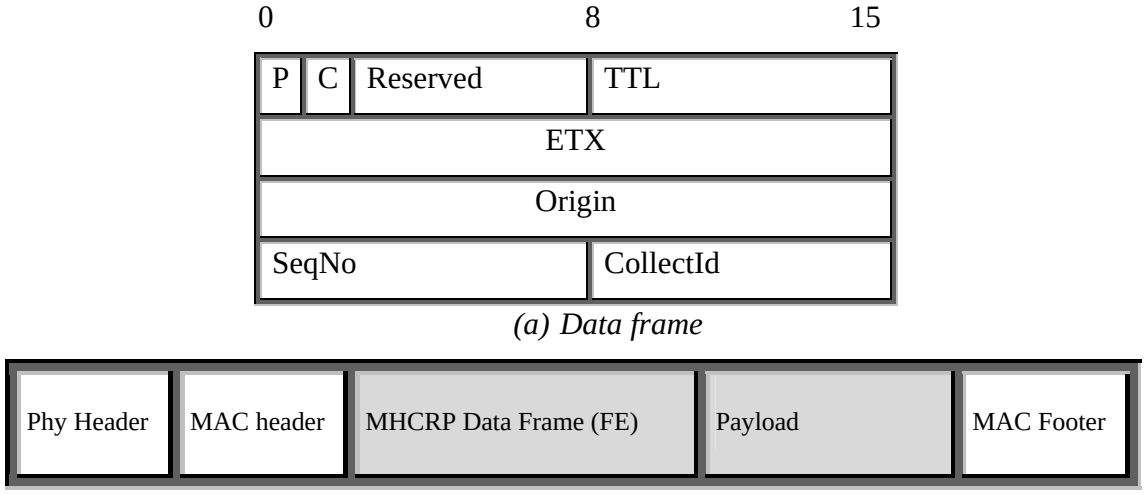
The sensor nodes using *MhcrpRoutingEngineC* module then send beacons at a random instant which is managed by a timer (*BeaconTimer*) to other nodes using *Beaconsend* interface. The other nodes which hear the beacon receive using an interface *beaconReceive* and update routing table using *routingTableUpdateEntry()* method. The information kept in the routing table helps nodes to determine which cluster head node or parent node to select in the multi-hop cluster tree formation.

As we described previously, *MhcrpRoutingEngineC* module also performs clustering the sensor nodes. This clustering process is triggered by timer, *ClusterTimer*. When *ClusterTimer* expires, each sensor node tries to elect itself as a cluster head using LEACH clustering technique as we described in Chapter 5. In this implementation, we have to define the number of cluster heads needed in the network before starting the clustering process. In our application, we require the number of cluster heads to be equal or greater than the number of categories of sensor nodes. When *ClusterTimer* expires, a node that was a cluster head then declares that it is not a cluster head (NON_CH) and it has no parent, then it invokes a method *joinClusterHead()* to join cluster head. This process is successful only if: the non cluster head node joins the cluster head node from those in its routing table that have valid path to the sink, is not congested, is not child of the

current node, lowest multi-hop ETX, and similar (same) in sensor type with the candidate cluster head node. If the node becomes a cluster head, it then again joins another node using a method *chooseParent()* to create multi hop cluster tree with the one that has valid path to the sink, is a cluster head, not congested, is not children of the cluster node, and a node with lowest multi-hop ETX. Finally, a routing update message is send to rapidly inform the neighbors.

Forwarding Engine Module: The main task of the Forwarding Engine consist in forwarding data packets which are received from neighboring nodes as well as sending packets generated by the application module of the node. In our implementation, in addition to the above mentioned tasks, Forwarding Engine module performs data aggregation operation where the cluster head intercepts incoming packets from its cluster members and apply MAX operator, selecting the maximum reading, on the intercepted packets before forwarding to the next hop.

For such purpose we define a module, *MhcpForwardingEngineC,* which performs the above mentioned tasks and define a data frame which is 8 bytes long as shown in Figure 6.4 (a).



(a) Data frame

(b) Data packet

Figure 6.4: Structure of MHCRP's (a) data frame and (b) data packet

The data frame is 8 bytes where the first two bits of the first byte include the P (Pull) and C (Congestion) flags. The former is used to trigger the sending of beacon frames from neighbors for topology update, while the latter allows a node to signal that it is congested. If a node receives a routing beacon from a neighbor with the C flag set, it will stop sending packets to this neighbor and look for alternative routes. The last 6 bits of the first byte are reserved for future use. The second byte reports the TTL (Time To Live) metric. The TTL is a counter that is

incremented by one at each packet forwarding and thus indicates the number of hops a packet has effectively traveled before reaching the current node. The third and fourth bytes are reserved for the multi-hop ETX metric, while the fifth and sixth constitute the Origin field, which includes the identifier of the node that originally sent the packet. The originating node also sets the 1-byte long SeqNo field, which specifies the sequence number of the packet. Further, the CollectId is an identifier specifying which application intended to handle the packet.

In our implementation, the cluster head using the *MhcpForwardingEngineC* module receives and queues the data packets to perform data aggregation. When a Timer (*aggregationTimer*) expires, the forwarding engine calls a method *aggregate ()* to perform aggregation and then use *send* interface to forward to the next hop in the multi-hop cluster tree.

Link Estimator Module: The Link Estimator is mainly responsible for determining the quality of the communication link between the nodes. We used the CTP implementation of LE, *LinkEstimatorP.nc* module, without modifying it in our implementation. The module performs the link estimation by counting the number of beacons actually received from each neighbor and comparing this number with the corresponding packet sequence number in the data frame, shown in Figure 6.4 (a), to determine the number of missed beacons which in turn help to determine the link quality between nodes. The detail operation how *LinkEstimatorC* module calculates the link quality between the sensor nodes using a metric called ETX is reviewed in Chapter 2.

6.3.3 LiveWatch: Data Logger Application

LiveWatch is a tool designed and implemented to receive data packet which is forwarded from the sensor network using Serial Forwarder (SF) tool and extract the data payload from other packet detail and then store the value in the database. We have implemented this tool using Java programming language and TinyOS library (net.tinyos.packet.*, net.tinyos.util.*, net.tinyos.message.*).

When the tool starts, it makes a connection with the MySQL database and then wait for an incoming packet from SF tool with the specified port number (for example, if the tool and SF are running on the same computer, the port number should be defined as sf@localhost:9001).

When the tool receives a packet from the SF, it then determines the packet length. If the length is equal with the expected packet length, then it parses the packet to get the sampled data by the

sensor node and stores it on the database. Otherwise ignore the packet and continue waiting for the next incoming packet.

The source code for this data logger implementation is found in Appendix B. Figure 6.5 shows the screen shoot of the data logger application.

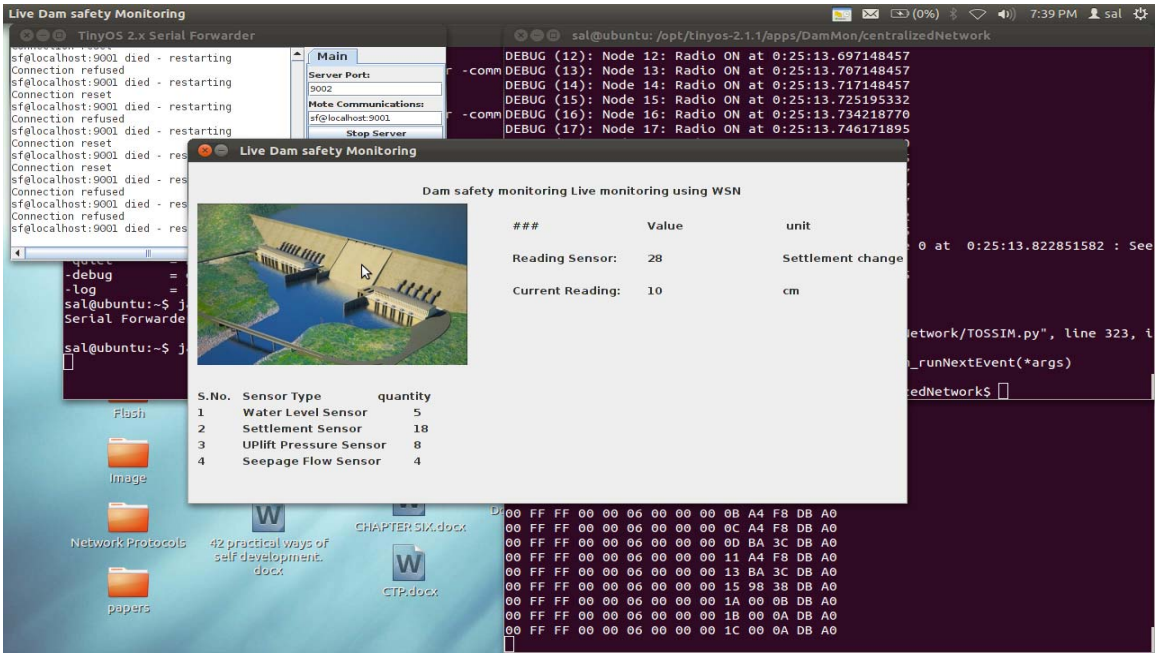


Figure 6.5: Data logger tool

6.3.4 Database Implementation

A centralized database is designed and implemented using MySQL database to hold the sampled data from the sensor network which is used for safety analysis and evaluation.

The information kept by the database consists of information about the deployed sensor nodes and their location, the reading value of safety parameter, geographical location of the sensor nodes and it is represented using relational database model.

Two main entities are identified after analyzing our dam safety monitoring requirement. The first entity, *NODE_INFO*, holds information about the sensor nodes like node identification number and type of parameter it measures whereas the second entity, *MEASUREMENT*, is a generalized entity that encapsulate *WATER_LEVEL_MEASUREMENT*, *UPLIFT_MEASUREMENT*, *SEEPAGE_MEASUREMENT*, and *SETTLEMENT_MEASUREMENT* that holds about the reading value of the sensor and the time of reading. In our implementation, we named the

database as, *DamSafeMonDB*, which contains five tables: NODE_INFO, WATER_LEVEL_DATA, UPLIFT DATA, SEEPAGE_DATA, and SETTLEMENT_DATA.

- NODE_INFO - contains information about the node.
- WATER_LEVEL_DATA - contains the measurement data done by the water level sensor nodes.
- UPLIFT_DATA - contains the measurement data done by the uplift sensor nodes.
- SEEPAGE_DATA - contains the measurement data done by the seepage sensor nodes.
- SETTLEMENT_DATA - contains the measurement data done by the settlement sensor nodes.

Figure 6.6 shows the detail design of the relational database for dam safety monitoring application.

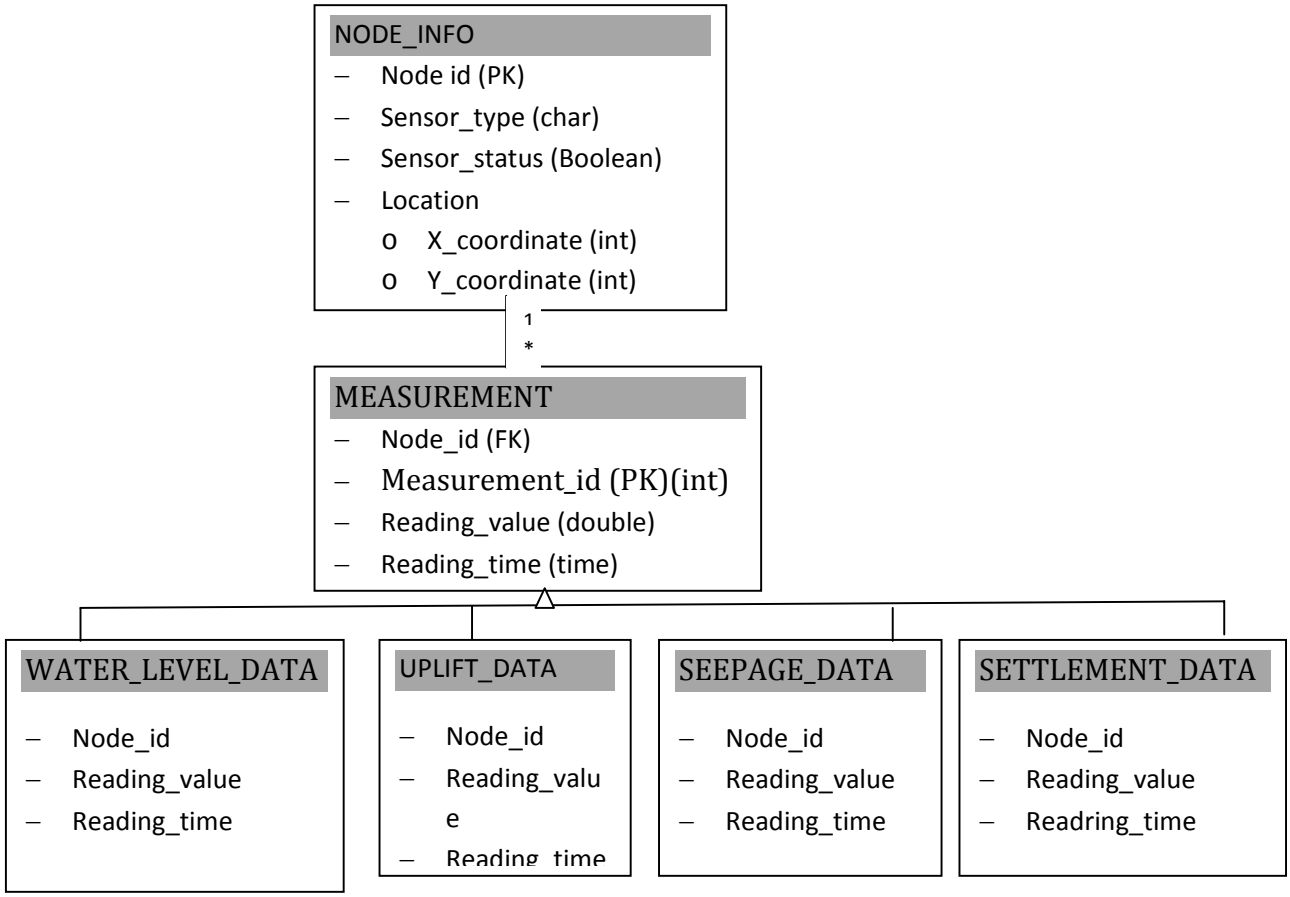


Figure 6.6: E-R diagram for sensor database

6.3.5 Web Application Implementation

A web application is designed and implemented to provide a user interface for the dam safety personnel. The web application reads the monitoring measurement from the database and

publishes the real time monitoring data to a web page. Moreover, it shows the geographic location of sensor nodes on the dam surface using map and list of unusual measurement made by the sensor nodes to give special attention to such events.

The following screen shots show the web interface designed for our dam monitoring application. We implemented the web application using PHP web programming language and runs on Apache web server.

Figure 6.7 shows the welcome screen of the web application of our dam safety monitoring application.

Figure 6.7: Welcome interface of the application

The user can views the measurements of water level, uplift, seepage, and settlement sampled by the sensor nodes for a given time interval as shown in Figure 6.8.

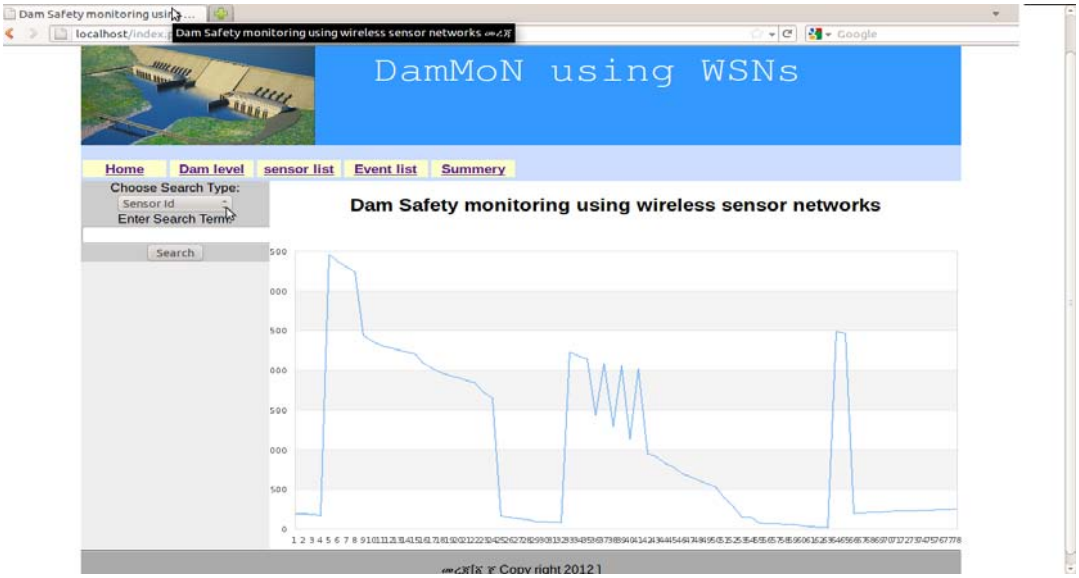


Figure 6.8: Water level, uplift, seepage, and settlement measurement

The user also can view the location of the sensor nodes on the dam surface using map as shown in Figure 6.9. This information helps the user to crosscheck the particular sensor reading with actual on-site inspection and also to take corrective action.

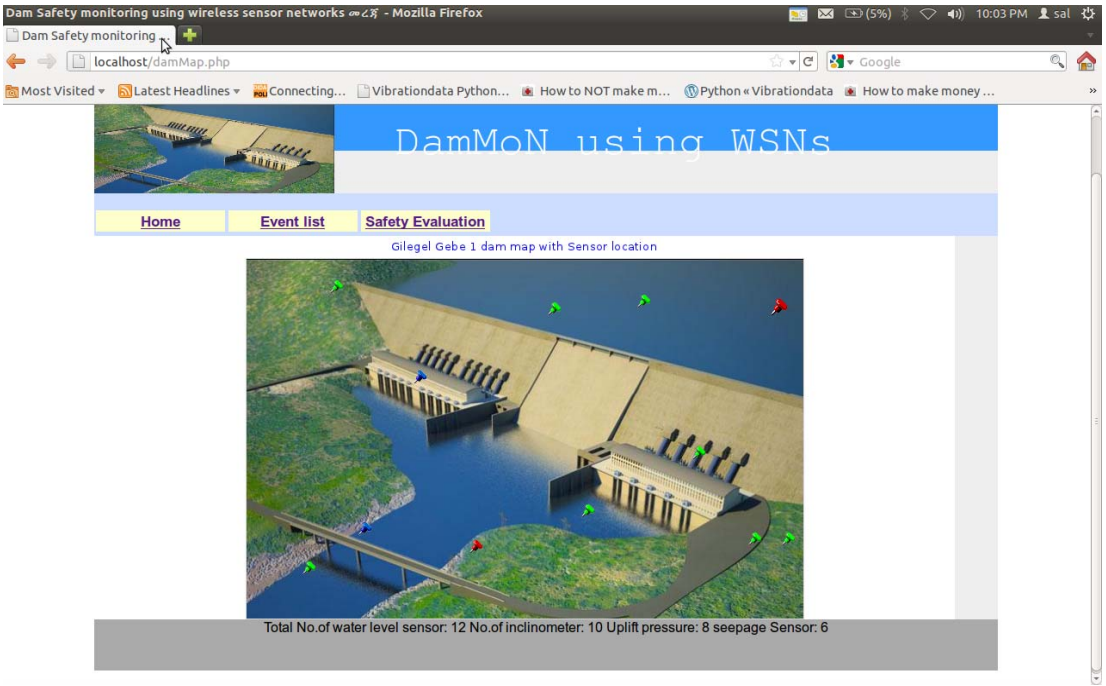


Figure 6.9: Dam map with the sensor node location

Figure 6.10 shows the major events that are recorded. The list shows the location, the time, and sensor which read the event.

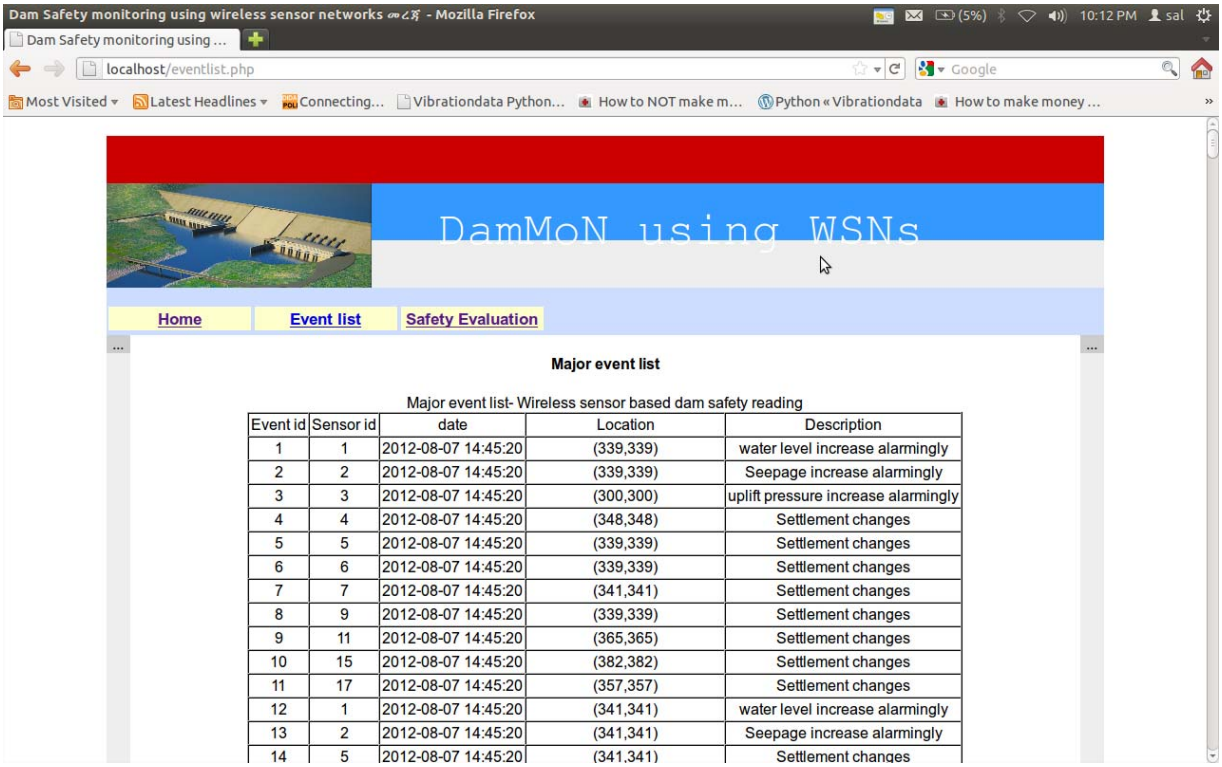


Figure 6.10: Event log

6.3.6 SMS Notification Implementation

The SMS notification tool is designed and implemented to provide a real time notification for the dam safety personnel using Python programming language, MySQLdb [34], and Python Serial [35] library. Our SMS notification application fetches data from the MySQL database through MySQLdb interface. To realize this implementation, Nokia 6300 model mobile phone is used and connected with the computer using USB cable. Mobile phone acts as a modem for the computer which runs the application to send SMS message in the GSM network. SMS notification tool use a port '/dev/ttyACM0', on Ubuntu machine and its baud (speed) set to 115200.

Figure 6.11: SMS notification: server running SMS app and user receive notification

Figure 6.11 shows the setup of the SMS notification tool while sending a notification message to dam safety personnel. The source code for this SMS implementation is found in Appendix C.

6.4 Simulation Experiment and Results

To test our wireless sensor network based dam safety monitoring application and investigate its performance, we performed a simulation experiment and evaluation using different metrics. To accomplish this we followed the following procedure: first, we defined the simulation setup where it encompass defining the arrangement of sensor nodes in the network, identifying the type of sensor nodes in the network, determining the channel access method used by the sensor nodes, channel model, and data traffic as described in Section 6.5.1. Second, we conducted computer simulation by deploying the different software tools that are developed for our dam safety monitoring application as described in Section 6.5.1. Lastly, we determined the evaluation metrics that help us observe and evaluate the behavior of the application in Section 6.5.3 and then we described our experimental analysis in Section 6.5.4.

6.4.1 Simulation Setup

Network Topology: We considered a typical embankment dam [43] of length of 700 meters and width of 300 meters and over which we assumed to deploy a total number of 35 nodes at the critical positions of the dam. Out of this total number of sensors: ten are settlement/alignment sensors, five are uplift pressure sensors, nine are reservoir level sensors, and ten are seepage

sensors. Again for simulation purpose we categorized sensors based on their sensor type. Let G_1 be set of sensor nodes which sense the reservoir level {1, 2, 3, 4, 5, 6, 7, 8, 9}, G_2 be the set of sensor nodes which sense uplift pressure {10, 11, 12, 13, 14}, G_3 be the set of sensors which sense seepage flow {15, 16, 17, 18, 19, 20, 21, 22, 23, 24}, and G_4 be the set of sensors which measure settlement movement {25, 26, 27, 28, 29, 30, 31, 32, 33, 34}. We have kept the position of the sensor nodes in a file, *topologyFile*, where each sensor node's position is represented as (x, y) coordinate as shown below.

```
1    700.0   200.0
2    623.0   200.0
3    546.0   200.0
…     …        …
35   750.0    150.0
```

The base-station sensor node (node 0) is located at far end of the dam at (750,150) as shown in Figure 6.12.

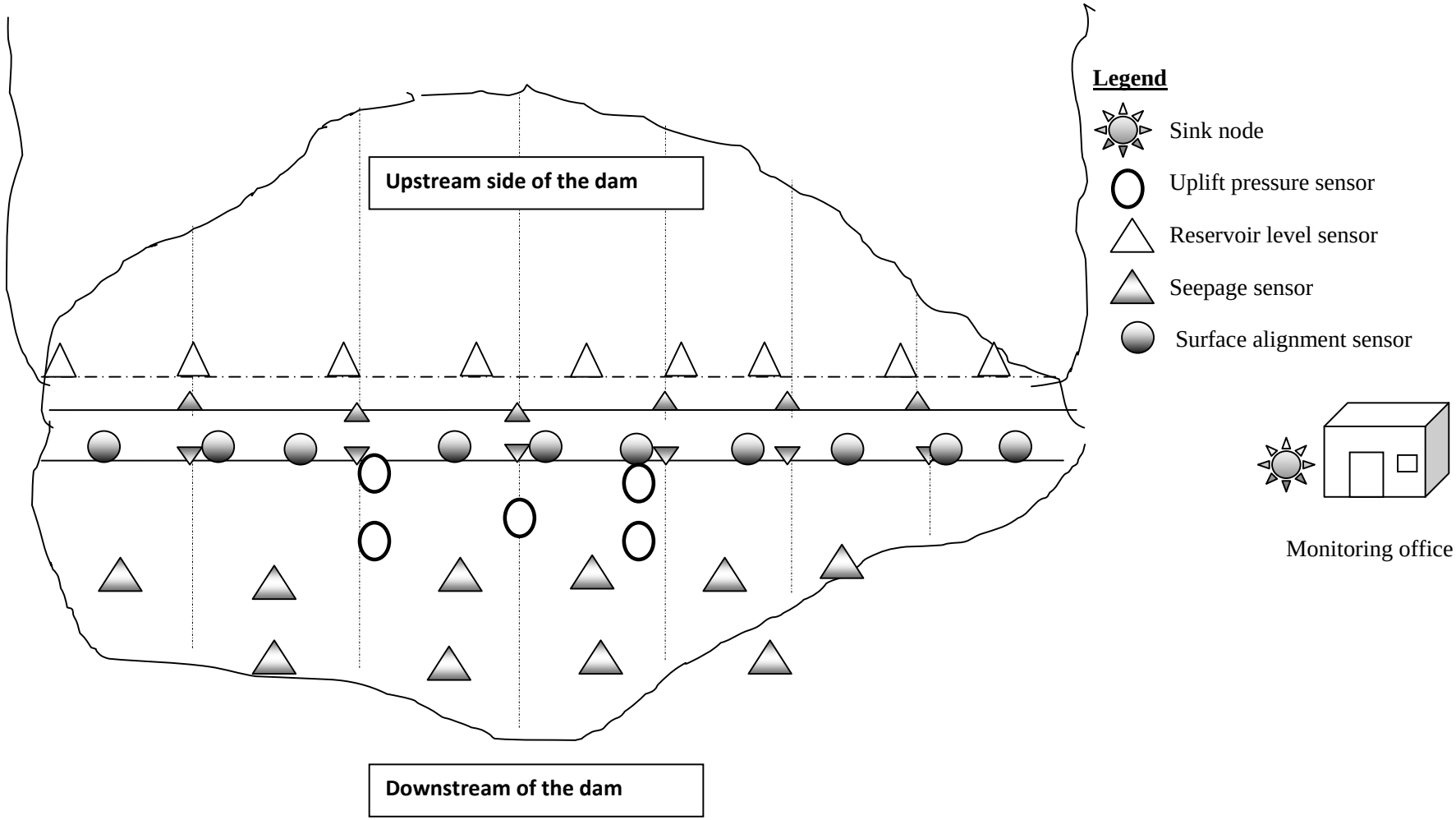


Figure 6.12: Wireless sensor network deployment layout on dam surface.

Link Layer Model: We have chosen IEEE 802.15.4 as a medium access control protocol because this protocol is designed to meet low-latency and low power requirement of wireless sensor networks and it is a standard for low rate wireless personal area networks. The default implementation of MAC protocol used by TOSSIM simulator is Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA). Thus, we employ CSMA-CA channel access mechanism in our simulation experiment.

Channel Model - In order to obtain reliable dam safety monitoring result, the radio link behavior between sensor nodes must be carefully studied. The behavior of a wireless link between the sensor nodes depends on the characteristics of the transceiver and where the sensor nodes are placed on. The channel is modeled using the log-normal path loss model [45] as shown below.

$$PL(d) = PL(d_0) + \eta \times 10\log_{10}(d/d_0) + X\sigma$$

In the equation above, $PL(d_0)$ represents the (known) path loss at a reference distance d_0, η is the path loss exponent, and $X\sigma$ is a gaussian zero-mean random variable with standard deviation σ. We define the following channel parameters based on the suggestion in [56] that characterizes our wireless sensor network channel in the dam environment:

η = 3.0;

SHADOWING_STANDARD_DEVIATION = 4.0;

d_0 = 1.0;

$PL(d_0)$ = 55.0;

Based on this parameter, we generate a link gain quality between wireless sensor nodes and kept in a file, *linkgain.out*, which specifies each link in a single line with three values, the source, the destination, and the gain, as shown below.

```
1   2    -54.0
2   1    -55.0
1   3    -60.0
3   1   -60.0
2   3   -64.0
3   2    -64.0
```

Data Traffic: In our application, the sensor nodes generate data traffic according to our adaptive sample timing technique, the default sampling frequency is fixed a priori and equal for all nodes

that are same in sensor type, sample the physical phenomenon (e.g., gather a reading from the water level sensor), and send the data to a central sink through a multi-hop cluster based routing tree using MHCRP protocol. Before the sensor node goes to sleep mode, the node determines the next sampling time based on the relative difference of two successive readings with the threshold value.

Physical Process: To simulate a physical process whose samples are collected and reported to the sink by the sensor nodes, we used a simulated sensor. In particular, demo sensor allows generating a sensor field at time t. The value of the source at time t is indicated as *V (t)* and since the actual values of the physical process are not critical for this study, we simply express the value of the sensor field at each time *t* using simple mathematical formula.

Table 6.1 summarizes the values we assigned to the simulation parameters in the context of our simulation study.

Table 6.1: Simulation parameters that are used for this work

Parameter	Value
No. of sensor nodes	35
No. of base-station node (sink node)	1
Simulation time	~990 seconds
Simulation area (m x m)	704 x 300
Node placement	According to the critical point of the dam
Outdoor Radio range	75 to 100m
Source data rate	250kbps
Transmit energy (TX)(mA)	17.4
Receive energy (RX)(mA)	19.7
Radio off (mA)	0
Node starting energy (mJ)	21600000

6.4.2 Simulation Process

Once the simulation setup is defined as described in the previous section, we have conducted the simulation experiment using two computers which run Ubuntu 11.10 operating system and are

connected together through serial cable. The 1st computer runs the wireless sensor network application using TOSSIM simulator and the 2nd computer runs SF tool, data logger application, database, and web application. The flow chart in Figure 6.13 shows the simulation process of TOSSIM simulator.

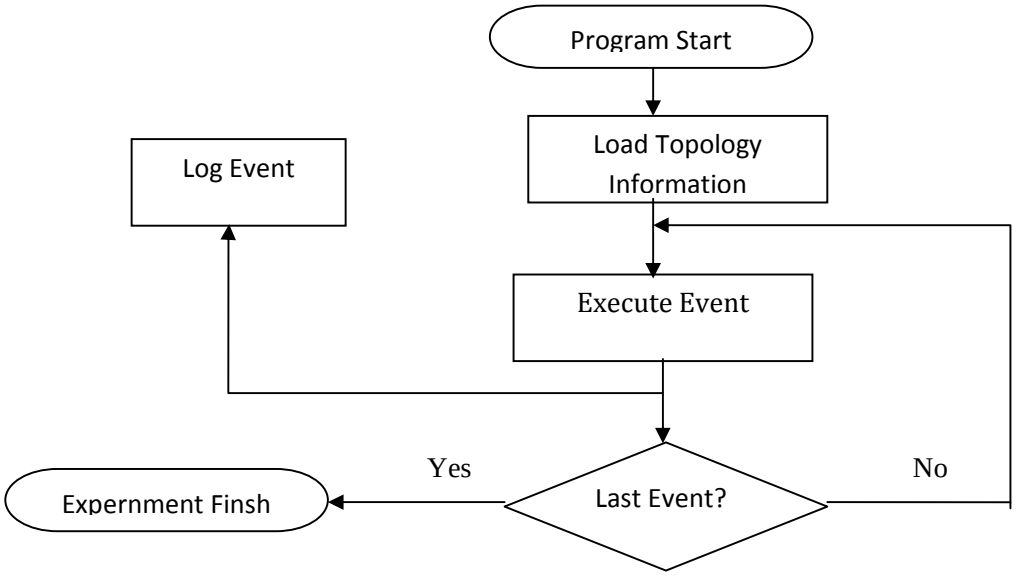


Figure 6.13: Simulation process flowchart

6.4.3 Evaluation Metrics

We evaluated the performance of our wireless sensor network based dam safety monitoring application and MHCRP routing protocol through a set of metrics that can summarize the most significant features of the application and compare its performance while it utilizes direct transmission, CTP routing protocol, and MHCRP routing protocol using the metrics described below.

Packet Delivery Ratio: It is the ratio of number of packets received at the destination to the number of packets sent by the source. Packet delivery ratio is an important metric as it describes the loss rate. For all our simulations we have kept the same traffic model, so the packet delivery ratio will give us a fair comparison as to how efficient the underlying routing algorithm is under similar traffic load and to choose the appropriate routing protocol.

Packet Delivery Ratio = number of packet received / number of packet sent

Energy Consumption: It is the average of consumed energy among all the nodes in the network. The power consumption is the power used by sensor nodes in the network, where it includes the

power consumption by radio communication, CPU computation, storage, and led light. The amount of power used during the simulation will be monitored and used for evaluating the protocols. Batteries have a finite amount of power and nodes die once power runs out. For this reason lower and uniform power usage across the network is preferable.

$$Total\ energy\ used = energy\ used\ by\ (CPU + Radio + EEPROM + LED)$$

Latency: This performance metric is used to measure the average end-to-end delay of data packet transmission. The end-to-end delay implies the average time taken between a packet initially sent by the source and the time for successfully receiving the message at the destination. This includes all the delays caused during route acquisition, buffering, and processing at intermediate nodes. This can be expressed as:

$$Latency = Time\,packet\,received - Time\,packet sent$$

Sample Rate: This performance metric is used to measure the rate at which sensor nodes make sample and communicated to the sink node. This metric help us to measure the number of missed events that are not recorded by the sensor nodes while utilizing periodic sampling and adaptive sampling techniques.

6.4.4 Performance Evaluations and Results

In this section, we discuss the performance evaluation of our wireless sensor network application and our application specific routing protocol, MHCRP. Finally, we report and comment experimental results related to the metrics introduced in the previous section, i.e., the packet delivery ratio, energy consumption by the sensor nodes, latency, and sample rate. The comparison made when the application utilized direct transmission, CTP and MHCRP routing protocols.

Packet Delivery Ratio (PDR): To evaluate the ability of our dam safety monitoring application to reliably report data to the sink node, we compute and compare the PDR achieved by direct transmission, CTP, and MHCRP routing protocol. In the case of direct transmission network configuration, the sensor nodes send their sample directly to the base station (sink) node. In the CTP case, nodes wake up, construct the routing tree and use it to report data to the sink. However in our MHRCP routing protocol, the nodes wake up, construct a multi-hop cluster tree

and use it to report to the sink. We collect the number of packets sent by the sensor in the network and the number of packets reached at the base station for selected nodes. Table 6.2 shows the number of packets sent and delivered to the sink by selected sensor nodes and its corresponding PDR value.

Table 6.2: Number of sent, reached packet to sink, and its PDR of selected sensor nodes

	Direct Transmission			**CTP**			**MHCRP**		
	packet sent	**packets reached at sink**	**PDR**	**packet sent**	**packets reached at sink**	**PDR**	**packet sent**	**packets reached to sink**	**PDR**
Node 1	104	80	0.76	109	85	*0.78*	90	70	0.77
Node 20	117	40	0.34	118	62	*0.52*	90	60	*0.66*
Node 33	130	23	0.17	105	50	*0.47*	90	60	*0.66*

Figure 6.14 shows the PDR of the selected sensor nodes when they run the dam safety monitoring application under direct transmission, CTP, and MHCRP protocol.

As can be seen in Figure 6.14, when our application utilizes MHCRP have shown a higher PDR value as compared with the direct transmission and CTP routing protocol. This shows that our application reliability transfers the monitoring sample data from the sensor nodes to the sink node even from those nodes which are far from the sink node.

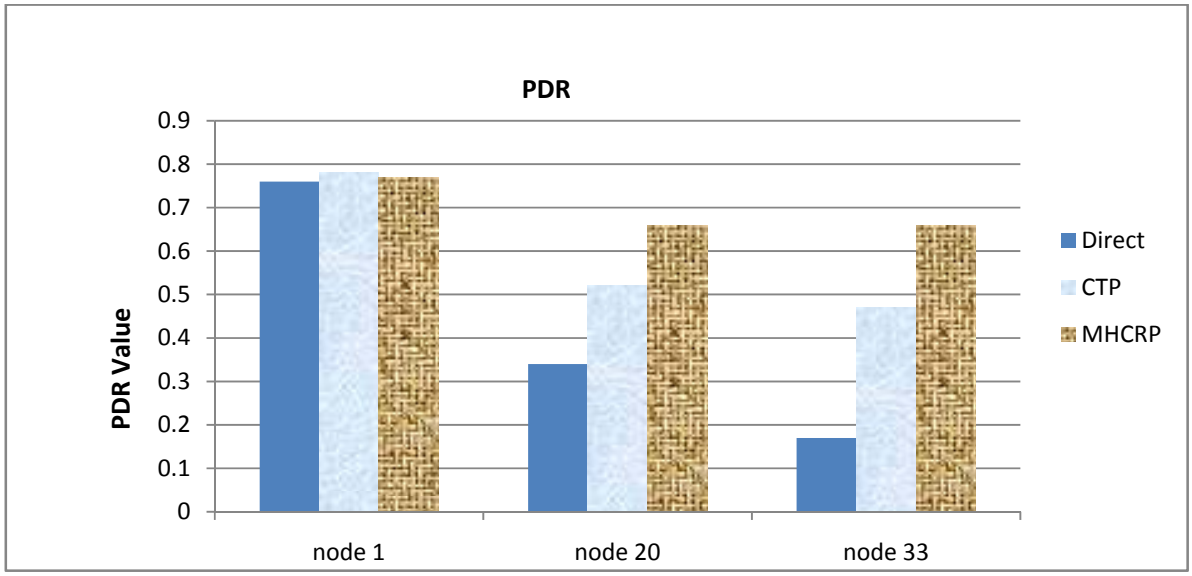


Figure 6.14: PDR of the application employing different routing protocols

Energy Consumption: To evaluate the energy consumption of sensor nodes, we assume that all the nodes in the simulation model starts out with a 2160000 mJ of energy. The node will be considered exhausted of energy when energy is zero. The energy expenditure by radio transmission and computation is high and used for evaluation of sensor nodes energy consumption in the network. The energy consumption by the sensor LED and memory is nearly zero. Table 6.3 shows the power consumption of selected sensor nodes in the network.

Table 6.3: Power consumption of selected sensor nodes on different routing protocol

	Direct Transmission				CTP				MHCRP			
	CPU	**Radio**	**Total energy used**	**Battery energy remaining**	**CPU**	**Radio**	**Total energy used**	**Battery energy remaining**	**CPU**	**Radio**	**Total energy used**	**Battery energy remaining**
Node 1	0.1	4310.4	4310	21595519	0.1	4309.8	4310	21595520	4.5	294757.5	294762	21293606
Node 20	0.1	4320.6	4321	21595508	0.1	4316.9	4317	21595513	4.5	294764.8	294769	21293598
Node 33	0.1	4328.1	4328	21595501	0.1	4326.9	4327	21595503	4.5	294763.9	294768	21293599

As can be seen in Figure 6.15, the sensor nodes consume more energy when they utilize MHCRP routing protocol, however, it is not desirable for wireless sensor network application. This is due to the fact that in MHCRP routing protocol the sensor nodes send many routing beacons and perform clustering operation which dissipate much energy.

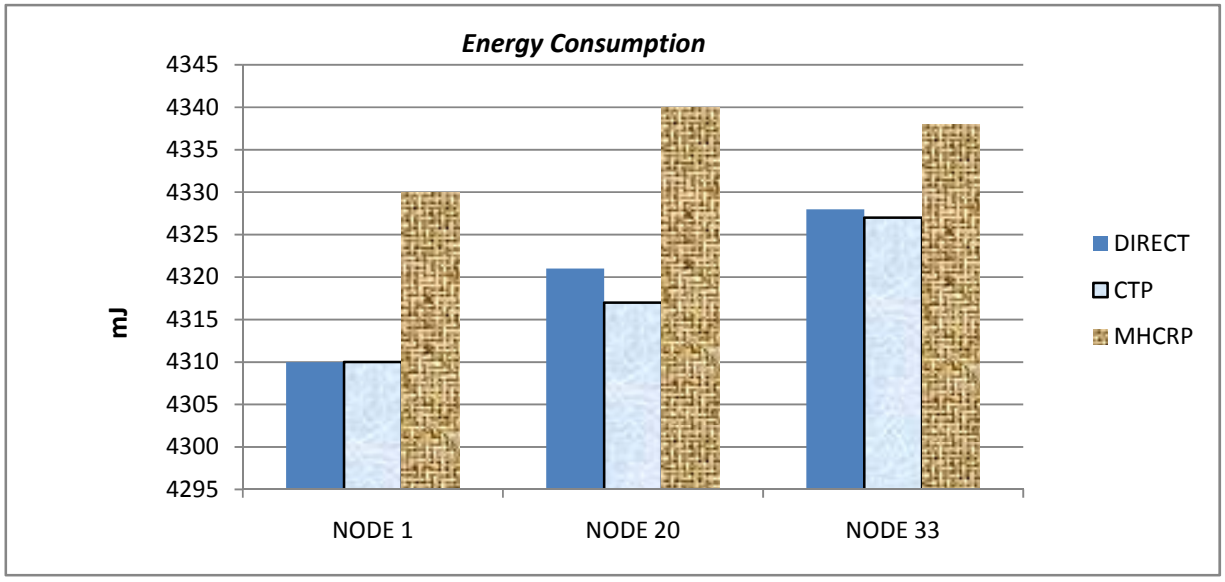


Figure 6.15: Energy consumption of nodes in direct transmission, CTP, and MHCRP routing protocol

Latency: In our simulation, we time stamp a packet when the sensor node starts sending data packet and also the time when that particular packet reached to the sink node for all packets transmitted for a single simulation experiment.

We showed the average end-to-end delay of selected sensor nodes in Figure 6.16. As can be seen from the graph, MHCRP has shown higher average end-to-end delay almost for all sensor nodes. This is due to the fact that the packet is buffered for certain amount of time in the cluster head node in each hop. However, this latency is tolerable for our dam safety monitoring application since our application is soft real time application where the dead line is not a requirement.

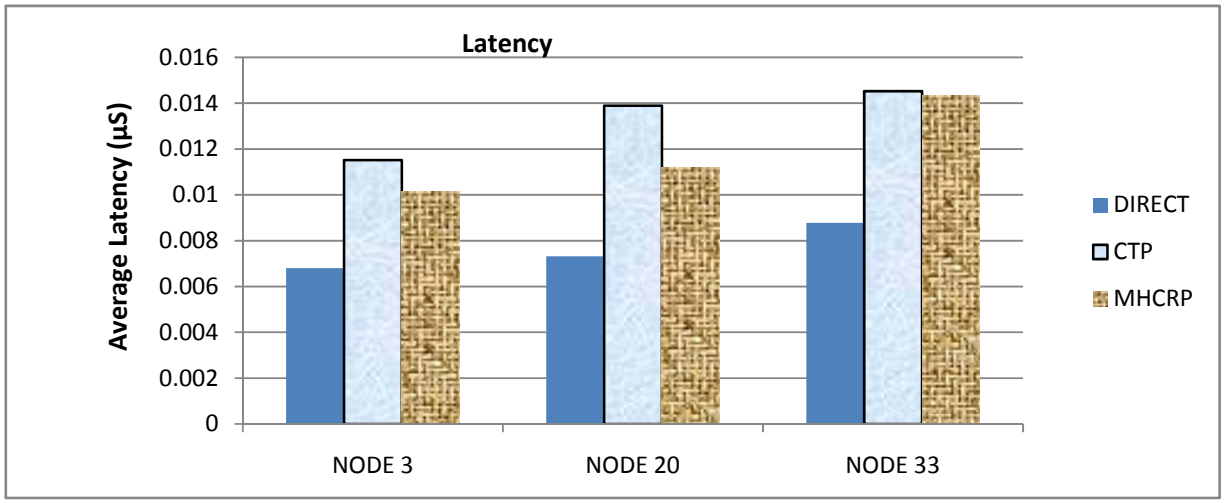


Figure 6.16: Average Latency in direct transmission, CTP, and MHCRP routing

Sample Rate: It is the rate that sensor data can be taken at each individual sensor and communicated to the sink node. We have collected sample data using periodic sampling technique and adaptive sampling technique, the techniques we proposed for our application, the nodes adjust their sampling frequency based on the difference between two successive samples. As can be seen in Figure 6.17, when the application utilizes adaptive sampling, it reduces the number of sampled data which was 30 in periodic sampling to 19 for a given period of monitoring period. This in turn reduces the energy consumption of the sensor nodes while achieving the desired monitoring task.

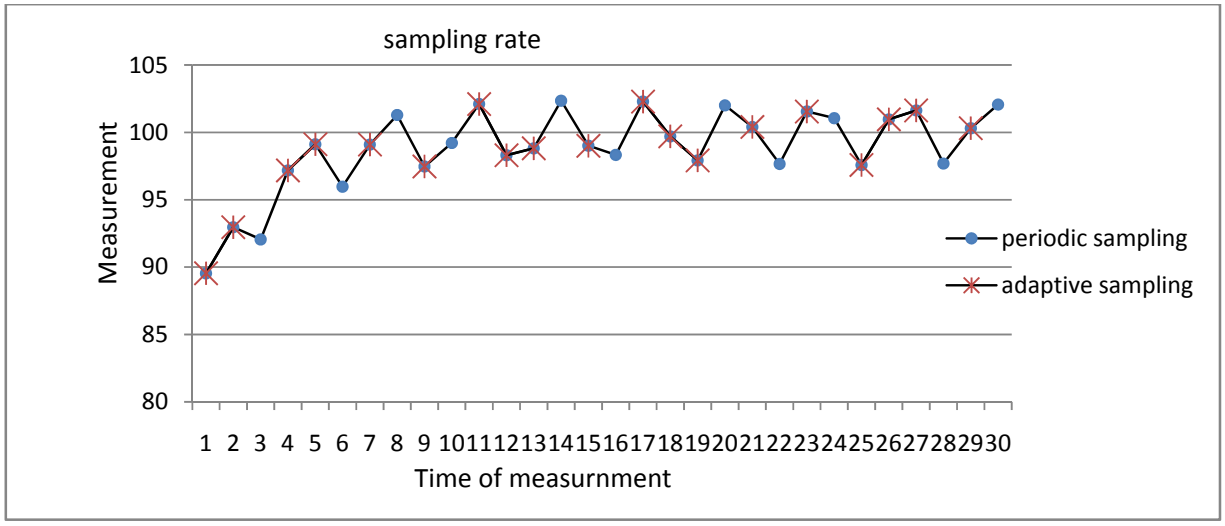


Figure 6.17: Sampling: Periodic sampling vs. Adaptive sampling

6.5 Summary

In this Chapter, we described the implementation details, simulation experiment, and evaluation result of our dam safety monitoring application. We implement the wireless sensor application for our dam safety monitoring and also application specific multi hop routing protocol. We also implemented different data management and visualization tools. To evaluate the performance of our application and application specific routing protocol, we performed simulation experiment using TOSSIM simulator. The simulation experiment result shows our application and routing protocol meet the dam safety requirement with high PDR and tolerable end to end delay.

CHAPTER SEVEN: CONCLUSION AND FUTURE WORKS

7.1 Conclusion

Dam is a critical civil infrastructure for many countries like Ethiopia, which has constructed a number of dams and continues constructing for the purpose of generating hydroelectric power and for irrigation. As it is a critical civil infrastructure, keeping track of dam safety by obtaining useful information about the behaviors of the dams, predict the future values, and detect abnormalities as early as possible to enable a timely response is the most important task. Traditionally dam safety monitoring is performed by examining the health of a dam by a team of persons who are qualified either by experience or education to evaluate a particular structure. It is based on a review of existing data and information, first hand input from the field and operational personnel, and site inspection. This is a labor intensive activity and may reduce the quality of the data gathered. As a result, the reliability and accuracy of the dam safety monitoring task will be doubtful. To monitor the safety of a dam in real time, it is necessary to adopt an automated monitoring system which increases dam monitoring frequency, decreases monitoring cost, reduces the probability of catastrophic failures, save lives, property, water storage and help for continuous monitoring of the dam in real time and predict the behavior of the dam more accurately.

To supplement the visual observations made during an inspection instruments like Piezometers, Inclinometers, Tiltmeters, Crackmeters, and Jointmeters are currently used. This instrumentation based monitoring method, however, requires cabling which is expensive. The cabling of a dam safety monitoring system is a complicated project, and the installation, adjustment and expansion of the monitoring points are also not sizeable. An alternative to dam safety monitoring system can be based upon wireless networks composed of inexpensive distributed sensor nodes. The size, low cost, and expandability of wireless sensor nodes make them the ideal candidate for monitoring of dam safety.

In this thesis, we proposed a dam safety monitoring system using a wireless sensor network that consists of wireless sensor nodes which are highly energy and processing power constraint devices, a technique for data collection, routing, and energy efficient data processing. We propose a model/framework for a system to perform automated safety monitoring of a dam using

a wireless sensor network by integrating the above mentioned techniques with the data management, safety analysis, and visualization techniques.

The system designed in this thesis contributions to the field of wireless sensor networks for the application of dam safety monitoring. The main accomplishments of this work are: modeled a framework for dam safety monitoring using wireless sensor network, developed a dam safety monitoring application running in MicaZ sensor node under TinyOS operating system that allows measuring of different parameters of the dam, developed an application specific clustering and routing protocol to route node's readings to a sink node. The routing and clustering protocol is developed considering the routing requirements of the dam safety monitoring application.

We have evaluated, analyzed, and proved that a dam safety monitoring system using a wireless sensor network and the application specific routing protocol, MHCRP, to have very good performance for delay and packet delivery when compared with centralized and CTP protocol. However, MHCRP has shown a significant power consumption compared to the other two routing protocols. This is because that MHCRP creates a multi-hop cluster where the nodes execute clustering protocol to determine the cluster head nodes. This protocol as shown from the simulation result and complex computing power, its power consumption is not effective when compared to others. As an output from this thesis, MHCRP satisfies most of the requirements for dam safety monitoring using wireless sensor network applications.

7.2 Future Works

Although we tried our best to realize the proposed architecture for dam safety monitoring application using a wireless sensor network with the objective of addressing the shortcomings of existing works, we do not believe that the architecture is generic enough to incorporate potential issues in dam safety monitoring. For instance, despite the importance of the issue, we have not considered the security aspect of the wireless sensor networks in our architecture since it was beyond the scope of this work. Therefore, we believe that the proposed architecture can be enhanced in such a way that the security of wireless sensor networks is taken into account.

We have also not considered data compression techniques to overcome the bandwidth limitation of a wireless sensor network again it was beyond the scope of this work. Therefore, we believe that the proposed application can be enhanced in such a way that the bandwidth limitation of wireless sensor networks is taken into account.

Our work focused on dam safety monitoring using a wireless sensor network specifically on reliable data collection, routing, data management, safety analysis, and visualization techniques. However, to predict and evaluate the safety of the dam, dam safety evaluation techniques can be integrated for overall dam safety monitoring and evaluation purpose. Evaluating the safety of the dam is important for the dam owner to see the current or predict the status of the dam. In such case, data fusion techniques can be applied to conclude whether the status of the dam is very unsafe, unsafe, relatively safe, safe, or very safe based on collected data by the wireless sensor nodes. Thus, we shall deal on this issue to make this work more complete.

The other line of improvement is regarding testing it on physical sensors to assess its practical significance. Hence, by taking this simulation as a platform, a more realistic implementation of the proposed architecture is another area of improvement so as to maximize its usability in a real dam setting in the future.

REFERENCES

[1] Dam Safety 101: Source: http://www.damsafety.org/news/?p=e4cda171-b510-4a91-aa30-067140346bb2, accessed on 18 September, 2011.

[2] TinyOS: http://www.tinyos.net/, accessed on 18 September, 2011.

[3] NesC: http://nescc.sourceforge.net/, accessed on 18 September, 2011.

[4] P. Levis, N. Lee, M. Welsh, and D. E. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications." In ACM Conf. on Embedded Networked Sensor Systems (SenSys), 2003.

[5] List of Dams in Ethiopia http://en.wikipedia.org/wiki/dams_and_hyderopower_in_ethiopia.html, accessed on 29 March, 2012.

[6] MicaZ Data Sheet: www.xbow.com, accessed on 18 September, 2011.

[7] TI CC2420 RF Transceiver - http://uk.farnell.com/texas-instruments/cc2420-rtb1/rf-transceiver-smd-qfn-48-2420/dp/1248485, accessed on 21 September, 2011.

[8] IEEE 802.15.4: http://www.ieee802.org/15/pub/TG4.html, accessed on 21 September, 2011.

[9] Yongfei Zhu and Chunlai Chai: "Sensor Network based Dam Safety Monitoring System," International conference on computer and Communication technologies in Agricultural Engineering, IEEE, 2010.

[10] Hill, J., "System Architecture for Wireless Sensor Networks." PhD Thesis, US Berkeley, 2003.

[11] Bo Xu, Congcong Tao, and Hui Xia, A Method for Dam Safety Evaluation based on Dempster-Shafer Theory, Proceedings of the advanced intelligent computing theories and applications, ACM 2010.

[12] Zhao Erfeng and Li Ziyang, "Multi-dimensional Model-based Database System Development about Dam Safety Monitoring," Proceedings of the 2008 International Symposium on Computational Intelligence and Design, IEEE Computer Society, Washington DC, 2008.

[13] Sukun Kim, David Culler, and James Demmel: Structural Health Monitoring Using Wireless Sensor Networks, 2004.

[14] Hans-Joachim Hof: “Applications of Sensor Networks. In Algorithms for Sensor and Ad Hoc Networks,” pp 1–20, 2007.

[15] CC2420 IEEE 802.15.4 / ZigBee-ready RF Transceiver, Texas instruments.

[16] Durisic, M.P., Tafa, Z., Dimic, G. and Milutinovic, V.: A Survey of Military Applications of Wireless Sensor Networks, IEEE, 2012.

[17] Khaled Reza, A.S.M Tariq, and S.M Mohsin Reza, “Microcontroller based automated water-level sensing and controlling: Design and Implementation Issue,” Proceeding of the world congress on Engineering and Computer science, San Francesco, USA, 2010.

[18] Xiuhong Li, xiao Cheng, Peng Gong, and Ke Yan, “Design and Implementation of Wireless Sensor Network based Remote Water-Level Monitoring System,” Sensors Journal, 2011.

[19] V. Bettzieche, Agent Based Dam Monitoring, British Dam Society, 13th Biennial conference, 2004.

[20] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks, ACM, 2007.

[21] Matteo Bianchi, “Health monitoring of Arch dams – recent development, international workshop on the present and future in health monitoring,” September 2000.

[22] Jerome P. Lynch, kenneth J. Loh: “A Summary Review of Wireless Sensors and Sensor Networks for Structural Health Monitoring,” 2006.

[23] Ayman Z. Faza, Sahra Sedigh-Ali: “A General Purpose Framework for Wireless Sensor Network Applications.” Proceedings of the 30th Annual International Computer Software and Applications Conference, ACM, 2006.

[24] Demirkol, I Ersoy, C. Alagoz,F.: “MAC Protocol for Wireless Sensor: A survey”, IEEE 2006.

[25] Bill Owens and Hal D. Simpson, “Dam Safety Manual,” State of Colorado, 2002.

[26] Muhammad Omer Farooq, Thomas Kunz, “Operating Systems for Wireless Sensor Networks: A Survey,” Sensors 2011.

[27] ANSI/IEEE 802.11 Standard, “Wireless LAN Medium Access control (MAC) and physical layer (PHY) specifications”, 1999.

[28] Wireless Sensors Networks Characteristics: http://en.wikipedia.org/wiki/wireless_sensor_network, accessed on 23/03/2012.

[29] Pressure Sensor: http://www.geokon.com accessed on 23/03/2012.

[30] US Federal Emergency Management Agency, http://www.fema.gov/ hazard/damfailure/, accessed on 29/03/2012.

[31] Elena F., Michele R., Jorg W. and Michele Z. “In-network Aggregation Techniques for Wireless Sensor Networks: A Survey”, IEEE, wireless communication, April 2007.

[32] Major Dam failures: http://en.wikipedia.org/wiki/Dam_failure, accessed on 23/03/2012.

[33] Venkata A. Kottapalli, Anne S. Kiremidjian, Jerome P. Lyncha, Ed Carryerb, Thomas W. Kenny, Kincho H. Law, and Ying Lei: Two-Tiered Wireless Sensor Network Architecture for Structural Health Monitoring, 2003.

[34] MySQLdb Module: http://sourceforge.net/projects/mysql-python, downloaded on 28/05/2012.

[35] Python Serial: http://sourceforge.net/projects/serial, downloaded on 14/02/2012.

[36] Xinying M., Jinkui C., Linghan Z., Jing Q., “Development of Wireless Sensor Networks for Dam Monitoring”, Journal of Information & Computational Science, 2012.

[37] Hui Liu, Zhijun Meng and Shuanghu Cui: A Wireless Sensor Network Prototype for Environmental Monitoring in Greenhouses, IEEE, September 2007.

[38] Tomonori N., Parya M., Kirill M., Mitsushi U., Noritoshi M., Masataka I., Gul A., Billie F. S., Jr.,Yozo F. and Ju-Won S., “Reliable Multi-Hop Communication for Structural Health Monitoring,” Smart Structures and Systems, Vol. 6, 2010.

[39] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson and Philip Levis. “Four-Bit Wireless Link Estimation.” In Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets VI), November 2007.

[40] Philip L., Neil P., David C., Scott S., “Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks.” In Proceedings of the First USENIX Conference on Networked Systems Design and Implementation (NSDI), 2004.

[41] Omprakash Gnawali, Rodrigo Fonsecakash, “Collection Tree Protocol,” SenSys 09, ACM 2009.

[42] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Micro-Sensor Networks,” in Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS-33), January 2000.

[43] Engineering Guidelines for the Evaluation of Hydropower Projects: http://www.ferc.gov/industries/hydropower/safety/guidelines/eng-guide.asp, accessed on 16/04/2012.

[44] JpGraph: http://jpgraph.net/, downloaded on 14/02/2012.

[45] T. S. Rappaport. Wireless Communications – Principles and Practice, Prentice Hall, Upper Saddle River, 2002.

[46] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. TEP 123: The Collection Tree Protocol, Aug. 2006.

[47] O. Younis and S. Fahmy, “HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad hoc Sensor Networks,” IEEE Transactions on Mobile Computing, 2004.

[48] Glassar S.D., Some Real-World Applications of Wireless Sensor Nodes, SPIE symposium on smart structures and materials, San Diego California, March 2004.

[49] Enrico P., Art O. C., Ricardo S. C., Meriel H., Ciaran M. G., PowerTOSSIM Z: Realistic Energy Modeling for Wireless Sensor Network Environments, ACM, 2008.

[50] Jamal N. A., Ahmed E. K., Routing Techniques in Wireless Sensor Networks: A Survey, 2004.

[51] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister: System Architecture Directions for Networked Sensors, ACM, 2000.

[52] S. Hoon and Kim, A Public Alert System For Dam Discharge Using Wireless Sensor Network, 29th Annual U.S. Society on Dams Conference, 2009.

[53] N. Xu, S. Rangwala, A. Broad, K. Chintalapudi, R. Govindan, D. Ganesan, D. Estrin, Wisden: A Wireless Sensor Network For Structural Monitoring, SenSys’04, November 3–5, 2004.

[54] M. Ramesh, S. Kumar, and P. Rangan, Wireless Sensor Network for Landslide Detection, 2009.

[55] T. Dinh, W. Hu, P. Sikka, P. Corke, L. Overs, S. Brosnan, Design and Deployment of a Remote Robust Sensor Network: Experiences from an Outdoor Water Quality Monitoring Network, 32nd IEEE Conference on Local Computer Networks, 2007.

[56] K. Sohrabi, B. Manriquez, and G. Pottie, Near Ground Wideband Channel Measurement, Vehicular Technology Conference IEEE, 1999.

[57] G. Hackmann, W. Gue, G.Yan, C. Lu, S. Dyke, Cyber- Physical Codesign of Distributed Structural Health Monitoring with Wireless Sensor Networks, ACM, 2010.

APPENDIX A: DamSafeMon

```
#include <Timer.h>
#include "DamSafeMon.h"   // declare the packet format

module SensorNodeC @safe() {
        uses interface Boot;
        uses interface Leds;
        uses interface Timer<TMilli> as Timer0;
        uses interface Timer<TMilli> as Timer1;
        uses interface Timer<TMilli> as Timer2;
        uses interface Timer<TMilli> as Timer3;

        //radio interfaces
        uses interface SplitControl as RadioControl;

        //routing interfaces
        uses interface StdControl as RoutingControl;
        uses interface StdControl as DisseminationControl;
        //uses interface DisseminationValue<uint32_t> as DisseminationPeriod;

        uses interface RootControl;
        uses interface ClusterControl;
        uses interface CollectionPacket;
        uses interface CtpInfo;
        uses interface CtpCongestion;
        uses interface Queue<message_t*>;
        uses interface Pool<message_t>;
        uses interface CollectionDebug;
        uses interface Send;

        uses interface Packet as RadioPacket; // interface to build the packet
        uses interface AMPacket as RadioAMPacket;
        uses interface AMSend as RadioSend[am_id_t id]; // interface to send
        uses interface Receive as RadioReceive[am_id_t id]; // interface to receive

        //sensing interfaces
        //uses interface Read<uint16_t> as ReadVibration;
        uses interface Read<uint16_t> as ReadWaterLevel;
        uses interface Read<uint16_t> as ReadUplift;
        uses interface Read<uint16_t> as ReadSeepage;
        uses interface Read<uint16_t> as ReadSettlement;
}
implementation {
        uint32_t samp_period = SAMPLING_PERIOD_MILLI_DEFAULT;
        uint32_t samp_period_timer = SAMPLING_PERIOD_MILLI_DEFAULT; //10000        uint16_t
check_period = CHECK_PERIOD_MILLI_DEFAULT; //500
        uint8_t msg_received = NULL_MESSAGE;  // default 0
        uint8_t resend_attempts = 0; // MAX_RESEND_ATTEMPTS = 2
        bool busy = FALSE;
        bool started = FALSE;
        bool radioBusy, radioFull;
        message_t pkt; // message data structure
        message_t btrpkt;
        uint16_t vibration;  // sensed data
```

```
	uint16_t curr_reading =0;  // current reading value
	uint16_t prev_reading =0;  // previous reading value
	uint16_t threshold =5;  // threshold value
	uint16_t seqno; // sequence number
	uint16_t rootItrator = 2;  // root counter
//------------------------- node boot----------------------------------------------------------------------------------
	event void Boot.booted(){
	if(TOS_NODE_ID != 0){
		dbg("Boot", "Booting Sensor Node %hu at %s\n", TOS_NODE_ID, sim_time_string());
		call RadioControl.start();
	}
	}
//-----------------------------------radio management code-----------------------------------------------------------
	event void RadioControl.startDone(error_t err){
	//if(TOS_NODE_ID != 0){
		dbg("Radio", "Node %hu: Radio ON at %s\n", TOS_NODE_ID, sim_time_string());
		if(err == SUCCESS){
			call RoutingControl.start();
			if (TOS_NODE_ID % 500 == 0){
				call RootControl.setRoot(); // make root head

			}

			if(started == TRUE){
	call Timer2.startOneShot(CHECK_PERIOD_MILLI_DEFAULT*(MAX_RESEND_ATTEMPTS+1));
			}else{
				started = TRUE;
			}
		}
		else {
			call RadioControl.start();
		}
	}
	event void RadioControl.stopDone(error_t err){
	if(TOS_NODE_ID != 0){
		if(err == SUCCESS){
			dbg("Radio", "Node %hu: Radio OFF at %s\n", TOS_NODE_ID, sim_time_string());
		}
		else {
			call RadioControl.stop();
		}
	}
	}
	//send code
	event void RadioSend.sendDone [am_id_t id] (message_t* msg, error_t error){
	if(TOS_NODE_ID != 0){
		if (&pkt == msg){
			busy = FALSE;
		}
	}
	}
	//end of send code

	event message_t* RadioReceive.receive [am_id_t id](message_t* msg, void* payload, uint8_t len) {
	if(TOS_NODE_ID != 0){
		if (len == sizeof(SinkToNodeMsg)) {
```

```
                        SinkToNodeMsg* btrpkt = (SinkToNodeMsg*)payload;
                        if(btrpkt -> message == SAMPLING_PERIOD_MESSAGE){

                                dbg("Radio", "Node %hu receiving new Sampling Period at %s: %hu \n",
TOS_NODE_ID,sim_time_string() ,btrpkt -> sampling_period);

                                samp_period = btrpkt -> sampling_period;
                                samp_period_timer = samp_period;
                                msg_received = SAMPLING_PERIOD_MESSAGE;
                                call Timer0.stop();
                                call Timer1.startOneShot(TOS_NODE_ID*10);

                        }
                }
                else if (len == sizeof(SinkToNodeResendDataMsg)){
                        SinkToNodeResendDataMsg* btrpkt = (SinkToNodeResendDataMsg*)payload;
                        if(btrpkt -> message == RESEND_DATA_MESSAGE){
                                dbg("Radio", "Node %hu receiving Resend Data Request\n", TOS_NODE_ID);
                                if((btrpkt -> msg_received[TOS_NODE_ID-1]) == FALSE){
                                        msg_received = RESEND_DATA_MESSAGE;
                                        call Timer1.startOneShot(TOS_NODE_ID*10);
                                }else{
                                        call Timer2.startOneShot(0);
                                }
                        }
                }
                else if (len == sizeof(SinkToNodeResendAckMsg)){
                        SinkToNodeResendAckMsg* btrpkt = (SinkToNodeResendAckMsg*)payload;

                        if(btrpkt -> message == RESEND_ACK_MESSAGE){

                                if((btrpkt -> msg_received[TOS_NODE_ID-1]) == FALSE){

                                        dbg("Radio", "Node %hu receiving Resend Sampling Period Ack at
%s: %hu \n", TOS_NODE_ID,sim_time_string() ,btrpkt -> sampling_period);
                                        samp_period = btrpkt -> sampling_period;
                                        samp_period_timer = samp_period;
                                        resend_attempts = btrpkt -> resend_attempts;
                                        msg_received = RESEND_ACK_MESSAGE;
                                        call Timer1.startOneShot(TOS_NODE_ID*10);
                                }else{
                                        call Timer2.startOneShot(0);
                                }
                        }
                }
                else if (len == sizeof(NodeToNodeMsg)){
                        NodeToNodeMsg* btrpkt = (NodeToNodeMsg*)payload;
                        dbg("Radio", "Node %hu receiving From node at %s: %hu \n",
TOS_NODE_ID,sim_time_string() ,btrpkt -> vibration);
                }

        }
        return msg;
        }

        event void Timer0.fired(){
```

```
	if(TOS_NODE_ID != 0 && TOS_NODE_ID <10){
		//if ((curr_reading - prev_reading) >= threshold)
			dbg("Radio", "current reading %hu greater than previous reading %hu threshold
\n",curr_reading,prev_reading);

		samp_period_timer = samp_period_timer - RADIO_PERIOD_MILLI_DEFAULT;
//RADIO_PERIOD_MILLI_DEFAULT = 5000,
		if(samp_period_timer == 0){
			samp_period_timer = samp_period;
			call ReadWaterLevel.read();
		}
	}
	else if(TOS_NODE_ID >= 10 && TOS_NODE_ID <15 ){
		samp_period_timer = samp_period_timer - RADIO_PERIOD_MILLI_DEFAULT;
		if(samp_period_timer == 0){
			samp_period_timer = samp_period;
			call ReadUplift.read();
		}
	}
	else if(TOS_NODE_ID >= 15 && TOS_NODE_ID <25 ){
		samp_period_timer = samp_period_timer - RADIO_PERIOD_MILLI_DEFAULT;
		if(samp_period_timer == 0){
			samp_period_timer = samp_period;
			call ReadSeepage.read();
		}
	}
	else if(TOS_NODE_ID >=25 && TOS_NODE_ID <=35 ){
		samp_period_timer = samp_period_timer - RADIO_PERIOD_MILLI_DEFAULT;
		if(samp_period_timer == 0){
			samp_period_timer = samp_period;
			call ReadSettlement.read();
		}
	}
	}

	event void Timer1.fired(){
	am_id_t id;
	if(TOS_NODE_ID != 0){

		if(!busy){
			if(msg_received == SAMPLING_PERIOD_MESSAGE){

NodeToSinkAckMsg* btrpkt= (NodeToSinkAckMsg*)(call RadioPacket.getPayload(&pkt, NULL));

			if (call RadioSend.send[id](0, &pkt, sizeof(NodeToSinkAckMsg)) == SUCCESS){
			dbg("Radio", "Node %hu sending ack at %s\n", TOS_NODE_ID, sim_time_string());
					busy = TRUE;
				}

					call Timer3.startPeriodicAt(call Timer0.getNow()-
100,RADIO_PERIOD_MILLI_DEFAULT);
					//Set a periodic timer to repeat every dt time units.
			}
			else if(msg_received == RESEND_DATA_MESSAGE){
				NodeToSinkVibrationMsg* btrpkt= (NodeToSinkVibrationMsg*)(call
RadioPacket.getPayload(&pkt, NULL));
```

```
                                btrpkt -> vibration = vibration;


                        }
                        else if(msg_received == RESEND_ACK_MESSAGE){

                                NodeToSinkAckMsg* btrpkt= (NodeToSinkAckMsg*)(call
RadioPacket.getPayload(&pkt, NULL));
                                if (call RadioSend.send [id](0, &pkt, sizeof(NodeToSinkAckMsg)) ==
SUCCESS){
                                        dbg("Radio", "Node %hu sending ack at %s\n", TOS_NODE_ID,
sim_time_string());
                                        busy = TRUE;
                                }

                                call Timer3.startPeriodicAt(call Timer0.getNow() -
(CHECK_PERIOD_MILLI_DEFAULT*(resend_attempts))-100, RADIO_PERIOD_MILLI_DEFAULT);
                        }

                }else{
                        dbg("Radio", "Node %hu: radio busy \n", TOS_NODE_ID);
                }
        }
        }
        event void Send.sendDone(message_t* m, error_t err) {

           busy = FALSE;
           dbg("SensorNodeC", "Send completed.\n");
        }
        //end of receive code

        event void Timer2.fired(){
        if(TOS_NODE_ID != 0){
                call RadioControl.stop();
        }
        }
        event void Timer3.fired(){
        if(TOS_NODE_ID != 0){
                call RadioControl.start();
                call Timer0.startOneShot(100);
        }
        }

        event void ReadWaterLevel.readDone(error_t result, uint16_t data){
        am_id_t id;
        if(TOS_NODE_ID != 0){
                if(result == SUCCESS){
                        vibration = data;
                        //dbg("Radio", "Node %hu sampled Water level at  %s : and level reached %hu \n",
TOS_NODE_ID, sim_time_string(),data);
                                if(!busy){
                                        NodeToNodeMsg* btrpkt= (NodeToNodeMsg*)(call
RadioPacket.getPayload(&pkt, NULL));

                                        uint16_t metric;
                                         am_addr_t parent = 0;
```

```
                                        call CtpInfo.getParent(&parent);
                                        call CtpInfo.getEtx(&metric);

                                        btrpkt->source = TOS_NODE_ID;
                                        btrpkt->seqno = seqno;
                                        btrpkt->vibration = data;
                                        btrpkt->parent = parent;
                                        btrpkt->hopcount = 0;
                                        btrpkt->metric = metric;

                                       if (call Send.send(&pkt, sizeof(NodeToNodeMsg)) == SUCCESS){
                                                dbg("Radio", "Node %hu sampled Water level at  %s : and
level reached %hu \n", TOS_NODE_ID, sim_time_string(),btrpkt -> vibration);
                                                prev_reading = curr_reading;
                                                curr_reading = btrpkt -> vibration;

                                                busy = TRUE;

                                        }
                                }else{
                                        dbg("Radio", "Node %hu: radio busy \n", TOS_NODE_ID);
                                }
                        }
                }
                }
                event void ReadUplift.readDone(error_t result, uint16_t data){
                am_id_t id;
                if(TOS_NODE_ID != 0){
                        if(result == SUCCESS){
                                vibration = data;
                                if(!busy){
                                        NodeToNodeMsg* btrpkt= (NodeToNodeMsg*)(call
RadioPacket.getPayload(&pkt, NULL));

                                       //btrpkt -> nodeid = TOS_NODE_ID;
                                       uint16_t metric;
                                        am_addr_t parent = 0;

                                        call CtpInfo.getParent(&parent);
                                        call CtpInfo.getEtx(&metric);

                                        btrpkt->source = TOS_NODE_ID;
                                        btrpkt->seqno = seqno;
                                        btrpkt->vibration = data;
                                        btrpkt->parent = parent;
                                        btrpkt->hopcount = 0;
                                        btrpkt->metric = metric;


                                       if (call Send.send(&pkt, sizeof(NodeToNodeMsg)) == SUCCESS){
                                                dbg("Radio", "Node %hu sampled uplift data at  %s : uplift
%hu \n", TOS_NODE_ID, sim_time_string(),btrpkt -> vibration);
                                                busy = TRUE;
                                       }
                                }else{
```

```
                                        dbg("Radio", "Node %hu: radio busy \n", TOS_NODE_ID);
                                }
                        }
                }
                }
                event void ReadSeepage.readDone(error_t result, uint16_t data){
                am_id_t id;
                if(TOS_NODE_ID != 0){
                        if(result == SUCCESS){
                                vibration = data;
                                if(!busy){
                                        NodeToNodeMsg* btrpkt= (NodeToNodeMsg*)(call
RadioPacket.getPayload(&pkt, NULL));

                                        uint16_t metric;
                                         am_addr_t parent = 0;

                                         call CtpInfo.getParent(&parent);
                                         call CtpInfo.getEtx(&metric);

                                         btrpkt->source = TOS_NODE_ID;
                                         btrpkt->seqno = seqno;
                                         btrpkt->vibration = data;
                                         btrpkt->parent = parent;
                                         btrpkt->hopcount = 0;
                                         btrpkt->metric = metric;
                                        if (call Send.send(&pkt, sizeof(NodeToNodeMsg)) == SUCCESS){
                                                dbg("Radio", "Node %hu sampled Seepage data at  %s :
Seepage %hu \n", TOS_NODE_ID,sim_time_string(),btrpkt -> vibration);
                                                busy = TRUE;
                                        }
                                }else{
                                        dbg("Radio", "Node %hu: radio busy \n", TOS_NODE_ID);
                                }
                        }
                }
                }
                event void ReadSettlement.readDone(error_t result, uint16_t data){
                am_id_t id;
                if(TOS_NODE_ID != 0){
                        if(result == SUCCESS){
                                vibration = data;

                                if(!busy){
                                        NodeToNodeMsg* btrpkt= (NodeToNodeMsg*)(call
RadioPacket.getPayload(&pkt, NULL));

                                        uint16_t metric;
                                         am_addr_t parent = 0;

                                         call CtpInfo.getParent(&parent);
                                         call CtpInfo.getEtx(&metric);

                                         btrpkt->source = TOS_NODE_ID;
                                         btrpkt->seqno = seqno;
                                         btrpkt->vibration = data;
```

```
                                        btrpkt->parent = parent;
                                        btrpkt->hopcount = 0;
                                        btrpkt->metric = metric;

                                       if (call Send.send(&pkt, sizeof(NodeToNodeMsg)) == SUCCESS){
                                                dbg("Radio", "Node %hu sampled Settlement data at  %s :
Settlement %hu \n", TOS_NODE_ID, sim_time_string(),btrpkt -> vibration);
                                                busy = TRUE;
                                        }
                                }else{
                                        dbg("Radio", "Node %hu: radio busy \n", TOS_NODE_ID);
                                }
                        }
                }
                }
}
```

APPENDIX B: LiveWatch

```
import java.io.*;
import javax.swing.*;
import javax.swing.JOptionPane;
import java.awt.*;
import java.awt.event.*;
import net.tinyos.packet.*;
import net.tinyos.util.*;
import net.tinyos.message.*;
import java.sql.*;
public class LiveWatch extends JFrame {

	public LiveWatch(){

	}
	public static void main(String args[]) throws IOException {
	int reading[];
	Connection conn = null;
	try {
	Class.forName("com.mysql.jdbc.Driver");
	conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/DamSafeMonDB","root","root");
	System.out.println("grv");
	System.out.println(conn);
	} catch (Exception e) {
	System.out.println("Error getting connection to " + "mysql");
	e.printStackTrace();
	}
	String source = null;
	PacketSource reader;
	if (args.length == 2 && args[0].equals("-comm")) {
	source = args[1];
	}
	else if (args.length > 0) {
		System.err.println("usage: java net.tinyos.tools.Listen [-comm PACKETSOURCE]");
		System.err.println("       (default packet source from MOTECOM environment variable)");
		System.exit(2);
	}
	if (source == null) {
			reader = BuildSource.makePacketSource();
	}
	else {
			reader = BuildSource.makePacketSource(source);
	}
	if (reader == null) {
		System.err.println("Invalid packet source (check your MOTECOM environment variable)");
		System.exit(2);
	}

	try {
		 LiveWatch app = new LiveWatch();
		 app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		 reader.open(PrintStreamMessenger.err);
```

```
                for (;;) {
                        byte[] packet = reader.readPacket();
                        int plength = packet.length;
                                int value = 0;
                                int sensorNo=0;
                                int reading_time=0;
                                for (int i = 8; i < 10; i++){
                                  sensorNo = (sensorNo << 8) + (packet[i] & 0xff);
                                  lblSensorId.setText(sensorNo +" ");
                                  if(sensorNo>=1 && sensorNo<5){
                                          lblSensorType.setText("Water level");
                                          lblUnit.setText("Water level");
                                  }
                                  else if(sensorNo>=5 && sensorNo<15){
                                          lblSensorType.setText("Uplift pressure");
                                          lblUnit.setText("pascal");
                                  }
                                  else if(sensorNo>=15 && sensorNo<25){
                                          lblSensorType.setText("seepage flow");
                                          lblUnit.setText("litre/second");
                                  }
                                  else {
                                          lblSensorType.setText("Settlement change");
                                          lblUnit.setText("cm");
                                  }
                              }
                                for (int i = 10; i < 12; i++){
                                  value = (value << 8) + (packet[i] & 0xff);
                                  lblReadingValue.setText(value +" ");

                              }
                            try{
                                          Statement st = conn.createStatement();
                                          String query = " insert into sensor (sensorid,x) values (?, ?)";
                                          // create the mysql insert preparedstatement
                                          PreparedStatement preparedStmt = conn.prepareStatement(query);
                                          preparedStmt.setInt (1, sensorNo);
                                          preparedStmt.setInt (2, value);
                                          //preparedStmt.setInt (3, value);

                                          // execute the preparedstatement
                                               preparedStmt.execute();
                                }
                                catch (SQLException s){
                                          System.out.println("SQL statement is not executed!");
                                          System.err.println(s.getMessage());
                                }
                        Dump.printPacket(System.out, packet); // Dump class (print tinyos messages)
net.tinyos.message.Dump
                        System.out.println(); // print new line
                        System.out.flush(); //
                }
        }
        catch (IOException e) { System.err.println("Error on " + reader.getName() + ": " + e); }
        }
}
```

APPENDIX C: SMS Notification Application

```
#!/usr/bin/env python
import MySQLdb as mdb
import sys
import serial
import time
con = None
total=0
while True:
        try:
        con = mdb.connect('localhost','root','root','DamSafeMonDB')
        with con:
                cur = con.cursor()
                #cur.execute("SELECT VERSION()")
                cur.execute("SELECT * FROM sensor")
                numrows = int(cur.rowcount)
                        #rows = cur.fetchall()
                        for i in range(numrows):
                        row = cur.fetchone()
                        #print row[0], row[1],row[2]
                        x = row[0]
                        total = total + x
                print total

        except mdb.Error, e:

                print "Error %d: %s" % (e.args[0],e.args[1])
                sys.exit(1)

        finally:

           if con:
        con.close()

        ser = serial.Serial('/dev/ttyACM0', 115200, timeout=1) # the port number
        #print ser
        ser.write('ATZ\r')
        ser.write('AT+CMGF=1\r') # mode of sms text mode
        ser.write('AT+CMGS="+251912274777"\r') # the mobile number that u send to
        ser.write('Urgent:Current reading reached'+str(total))
        ser.write(chr(26))
        line = ser.readline()
        print line
        ser.close()
        time.sleep(5) # time to delay the
```

Declaration

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

Name: Salahadin Seid

Signature: _____

Place: Addis Ababa

Date: _____

This thesis has been submitted for examination with my approval as a university advisor.

Name: Dr. Mulugeta Libsie

Signature: _____

Place: Addis Ababa

Date: _____