Keio University Bachelor's Thesis Academic Year 2011

LEAF: A Library for Multi-Application Wireless Sensor Networks

Keio University Faculty of Environment and Information Studies

Nguyen Gia

Abstract of Bachelor's Thesis Academic Year 2011

LEAF: A Library for Multi-Application Wireless Sensor Networks

Summary

The Wireless Sensor Network (WSN) is an active research area. Its results have been usefully employed in many applications such as environment monitoring, embedded systems, surveillance, etc. The traditional WSNs are designed to run only one application in one sensor network. However, in many scenarios, to reduce the administrative cost and deployment cost of WSNs, it is desirable to use single sensor network for multiple concurrent applications.

This thesis proposes LEAF, a library for Multi-Application Wireless Sensor Networks (MA-WSNs). It contains high flexibility and high efficiency sensor nodes control protocols to adapt to the variety of requests in MA-WSNs. We have evaluated LEAF in the perspective of packet delivery ratio, number of packet restransmission, packet delay variation, energy consumption and storage usage in many test cases. Through careful evaluation, we confirmed LEAF can send data to applications with the packet delivery ratio of 99%, which is 37% better than the combination of CTP+Drip, two widely used protocol in current wireless sensor networks. Beside, LEAF can transmit data with the number of packet retransmission of 0.13 compare to 2.95 of related works. LEAF also can send data to applications with the deviation about 3% of the required packet interval. Through this results, we can conclude that LEAF fulfills the requirements of MA-WSNs and totally can be used as a solution of management problem in WA-WSNs.

Keywords:

1 Multi-Hop Routing   2 Data Collection   3 Data Type Control

4 Sample Rate Control   5 Multi-Application Wireless Sensor Networks

Keio University Faculty of Environment and Information Studies

Nguyen Gia

卒業論文要旨 2011 年度（平成２３年度）

LEAF: A Library for Multi-Application Wireless Sensor Networks

論文要旨

近年，技術の発展により，無線センサネットワーク (WSN: Wireless Sensor Network) に関する研究が増えてきている．それに伴い，環境モニタリングやユーザ監視など，WSN を利用したアプリケー ションが数多く提案，実装されている．従来の WSN の設計では，一つの WSN で一つのアプリケーションのみを実行できるよ うになっていた．一方，多くの環境において，複数のアプリケーションが同時に一つの WSN を利用 することで，ネットワークの構築や管理に要するコストを削減できることが分 かっている．

本論文では，単一の WSN を複数のアプリケーションが同時に利用する，マルチア プリケーション WSN (MA-WSNs: Multi-Application Wireless Sensor Networks) を実現するライブラリ，LEAF を提案する．LEAF は，MA-WSNs の様々な要求に対応するため，高い柔軟性とデータ収集効率を 提供する．本論文では LEAF を実環境において，パケット配送率，パケット転送回数，パケット遅延変動，電力消費，ストレージ使用量を評価した．評価を通じ，LEAF が既存の Collection Tree Protocol と Drip よりも，パケット配送率で 37%向上し，平均で 99%達成した事が実証された．また，1 パケット当たりの再送回数を 2.95 から 0.13 に削減した事が示された．これらのことから，LEAF が MA-WSNs の要求を満たすことが証明された．

キーワード:
1 Multi-Hop Routing　2 Data Collection　3 Data Type Control
4 Sample Rate Control　5 Multi-Application Wireless Sensor Networks

慶應義塾大学　環境情報学部
グエン・ザー

Contents

List of Figures

List of Tables

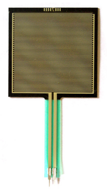
Chapter 1

Introduction

"Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user. " - *Mark Weiser* [1]

1.1 Background

A sensor is a device that measures a physical quantity such as temperature, humidity, light... and converts it into a signal which can be read by an observer or by an instrument. For example, a thermocouple (Figure 1.1(b)) converts temperature to an output voltage which can be read by a voltmeter. The advances in electrical engineering, especially, in micro electro-mechanical system (MEMS) have lead to the miniaturization of sensors. In addition, with the growth of wireless technologies, sensors have not only become smaller, but also have the wireless communication ability. As a result, Wireless Sensor Network(WSN), a network that is built from nodes - from a few to several hundreds or even thousands, where each node is connected to one or more sensors has been introduced and widely studied. Nowadays, sensors are used in everyday objects such as sensitive door, lamps or air conditioner. Also, there are innumerable applications of wireless sensor networks(WSNs), such as environment monitoring[2][3], surveillance[4][5], medication management[6][7], and so on. All of these bring convenience and ease to user, as ubiquitous computing is defined as *"machines that fit the human environment instead of forcing humans to enter theirs"*[8].


(a) Pressure Sensor

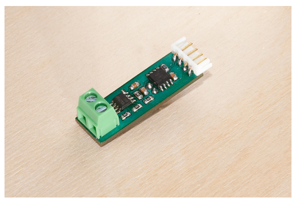

(b) Thermocouple

Figure 1.1: Some of sensors

1.2 Motivation and Objective

In current WSNs, commonly, all of sensor nodes are participating in a single global task. In other words, only one application can run on a single WSN. All of sensor nodes work in same sample rate, sense sample kind of data and transmit data to a special node called sink node. To use multiple application, we have to build multiple WSNs. However, in many scenarios, it will be desirable to use a single WSN for multiple concurrent applications to reduce the administrated cost and deployment cost of WSN. By this way, the return-of-investment will also be increased because of the usefulness of WSNs. So there is a need of an infrastructure or a management method to support Multi-Application Wireless Sensor Networks (MA-WSNs).

In the age of "Internet of Things" or "Global Wireless Sensor Networks", the number of sensor node in sensor networks will become very large and sensor will be deployed everywhere around us. In the future, by integrating sensor networks into the Internet, an unique global wireless sensor network can be created. At that time, a sensor node will be requested for data not only by local applications but also by global applications. For example, consider a mobile application that can send request to some of sensors in 5km radius circle from it. In this case, requested sensor will be changed depending on user's position. Towards a flexible global sensing infrastructure, firstly, the local multi-purpose WSNs need to be built.

TinyOS is an open source, component-based operating system designed for low-power wireless devices used in sensor networks. With Contiki, TinyOS is one of the most famous and widely used operating systems targeting WSNs. However, in the current time, the data collection protocols used in TinyOS only support specific purpose WSNs. To control sensor nodes in

WSN, TinyOS use the dissemination protocol that transmit control message to all of sensor nodes, which result in large packet transmission overhead, energy wastage and interference in network. Since sensor nodes are strictly constrained in on-board energy supply, an effective management method is necessary for TinyOS to support MA-WSNs.

This research proposes a flexible management method integrated into TinyOS to control sensor nodes as well as sensed data to support multi-application wireless sensor network. By creating effective sensor control protocols it provides robustness, energy efficiency, scalability to MA-WSNs.

1.3 Organization

This thesis is organized as follows.

In chapter 2, we will describe the traditional WSNs and through it's limitations, we will show why we need the MA-WSNs. We will also describe the concept, benefits, requirements and some related infrastructures of MA-WSNs. The data collection protocol, control protocol and the relationship among them will be introduced in chapter 3. Chapter 4 is about the approach and design, where we will describe the overall system and protocols that is used in it. The implementations of the system will be described in chapter 5. Chapter 6 show how we evaluated this system and results that we achieved. Finally, some of future works and conclusion will be stated in chapter 7.

Chapter 2

Wireless Sensor Networks

This chapter introduces the Wireless Sensor Networks. Firstly, the traditional Wireless Sensor Network and its limitations will be described. After that, the Multi-Application Wireless Sensor Network and its requirements will be illustrated. We will also represent some of related works in this chapter.

2.1 Traditional wireless sensor network

2.1.1 Background

A Wireless Sensor Network (WSNs) is a network built from sensor nodes. It consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, humidity, light, sound, pressure, vibration, motion or pollutants and to cooperatively pass their data through the wireless network to a main location. From the applications that used in millitary such as battlefield surveillance, today WSNs have been used in many industrial and consumer applications.

The typical WSNs architecture is desribed in figure 2.1. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The number of sensor nodes and the kind of sensed data is variety depend on the application. A forest fire detection WSN can have hundreds of temperature sensor nodes while a room's temperature monitor application only need a few ones. Each of sensor node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit

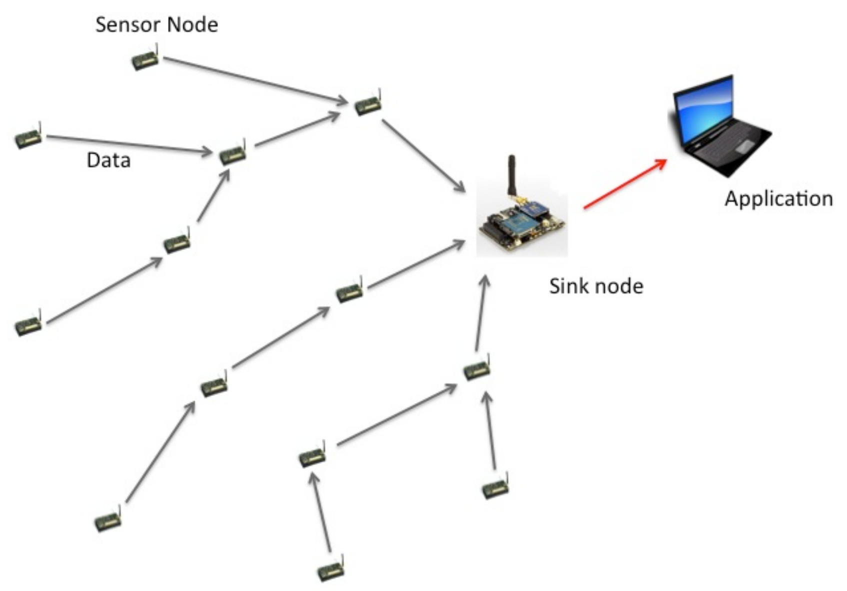


Figure 2.1: Typical WSN architecture

for interfacing with the sensors and an energy source. A sensor node can be connected to one or some of sensors which sense different kind of data. For example, SunSPOT[9] (figure 2.2(a)) sensor node includes temperature, light and acceleration sensor while μPart[10] (figure 2.2(b)) sensor node have light, tilt, temperature, motion and acceleration sensor. All sensor node periodically sense and transmit data to a special node called sink node. At sink node, data will be aggregated before sent to application in a terminal device such as a computer. Commonly, in tradition WSNs, sample rate and kinds of sensed data are configured in deploy process and not be changed at runtime.

2.1.2 Limitations of traditional wireless sensor network

In traditional WSNs, all of sensor nodes are participating in a single global task. In some scenarios, we need to run multiple applications at the same location. For example, in a smart house, we need to run the room's temperature monitor application and the thief detection application concurrently.

(a) SunSPOT

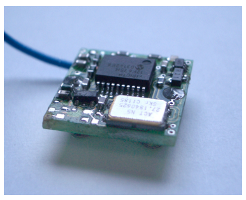

(b) μPart

Figure 2.2: Examples of sensor node

Since only one application can be run on a WSN, we have to build multiple WSNs to run multiple applications. It affects the usefulness and the cost of WSN.

In current WSNs, sample rate and the type of data are configured in deploy process and cannot be changed individually. Since all of sensor node work in same sample rate and sense sample kind of data, the sink node control sensor nodes by using dissemination protocols. The control message will be sent to all of sensor nodes. Although when only one sensor node's sample rate need to be changed, all of sensor node's sample rate will be changed to appropriate value. For example, all of sensor nodes are working with sample rate of 0.5Hz. Now, one of sensor nodes need to work at 1Hz. With current WSNs, all of sensor node have to change sample rate to 1Hz. It causes the large packet transmission overhead, the packet interference inside the network, data redundancy and energy wastage of WSNs.

In the future, WSNs tend to be integrated into the Internet to become the global wireless sensor network. Sensor nodes not only will be used by local applications but also used by global applications. Especially, with the explosion of mobile devices, there will be many mobile applications that request data from dynamic source depending on their locations. Therefore, a sensor node can receive external request at any unpredictable time. Since the traditional WSNs cannot serve these kinds of requests, at this time, we cannot build the global sensing infrastructure.

2.2 Multi-application wireless sensor network

2.2.1 Concept and target environments

This research provides a model of multi-application wireless sensor network (MA-WSN). It is a WSN that multiple applications can concurrently run on. Moreover, as a part of global wireless sensor network, it not only provide data to local applications but also can serve data request from external applications.

This model of MA-WSNs can be used in various scenarios with different conditions. From the home or office sensor network with applications like temperature monitoring, thief detection, user recognition application to the industrial factory sensor network with applications like structural monitoring, machine monitoring, this kind of MA-WSN is still useful and can be deployed.

2.2.2 Benefits

Firstly, by supporting multiple applications in one WSN, the administrated cost and deployment cost of WSNs will be reduced. We spend time, effort, and money to deploy only one WSN and then, we can run on it various concurent applications. Because the WSNs become more useful, the return-of-investiment will also be increased.

This model use flexible data collection protocol and effective control protocol that can dynamically change the type of sense data and the sample rate of sensor based on applications's request, which results in the roubustness of WSNs. Only necessary data are transmitted. The sensor nodes are scheduled to only sense and transmit data when required. In additon, instead of dissemination, the control message will be sent directly to the approriate sensor node. By this way, the packet transmission overhead, the packet interference, data redundancy, energy consumption will be decreased.

By providing the response to the external request, this type of WSNs can become part of a very large WSN, the global wireless sensor network. The WSNs applications can be used in everywhere at anytime. We can control home appliance when we are in company or school, a biologist can do research about Amazon rainforest while still living in Japan and many mobile applications can be simply created.

2.2.3 Requirements

An effective and useful MA-WSN have some of requirements that are explained as follows.

- A high flexibility data collection

  With the rapid development of electrical engineering, sensor nodes tend to be single devices which many kinds of sensor are integrated in. In other words, a single sensor node can sense several kinds of data. But it does not mean that all type of sensed data are used. Based on requests of applications, the data collection protocol have to dynamically choose appropriate data to collect.

- A high efficiency sensor control method

  In MA-WSNs, a sensor node is requested for data by multiple applications. Of course, requests from multiple applications are different in terms of sample rate and type of data. In addition, sensor nodes do not only receive request from local applications but also external applications. This kind of request can come at every unpredictatble time. Therefore, to reduce the packet transmission overhead and energy consumption but still guarantee accurate reponse to applications, sensor nodes need to be scheduled to sense and transmit data at right time.

- A efficient routing scheme

  To decrease the packet transmision overhead and the packet interference inside the network, MA-WSNs send control message directly to appropriate sensor node. Since almost energy of sensor nodes are used for radio transmission, we need to build a efficient routing protocol in terms of loss rate and hop number. The routing protocol not only used to select path from sensor node to sink node to send sensed data but also used to selected path from sink node to sensor node to send control message.

2.3 Related works

Related to MA-WSNs, some researchers have created some of infrastructured to support concurent multiple applications in one sensor network.

- Supporting concurrent applications in wireless sensor networks

  Yang Yu et al. have built Melete[11], a system that supports concurrent applications in a single wireless sensor network. Their work is based on the Mat virtual machine with some modifications and enhancements. They used the dynamic grouping based on contemporary status of the sensor nodes. The grouping procedure itself is programmed with the TinyScript language. A group-keyed code dissemination mechanism is also developed. In this research, they only consider a sensor network connected to a gateway where grouping instructions and application code are injected. Melete system only work on a sensor network that fulfills these requirements:

  (a) The network is connected
  (b) The gateway stores code for all applications
  (c) All sensor nodes support omni-directional broadcast

  In common, the sensor network’s topology is not a connected graph so this research result cannot widely be applied in real time deployments.

- Middleware for Resource Sharing in Multi-purpose Wireless Sensor Networks

  Pedro Javier del Cid et al. have developed a middleware for resource sharing in multi-purpose wireless sensor networks[12]. They consider the infrastructure of multi-purpose WSNs as a lightweight service platform that can provide services for multiple concurrent distributed applications. In this context, their research focuses on efficiently managing shared resources while considering Quality of Data and context aware operation. They addressed the issue of how concurrent use of WSN services may lead to consequential contention over a sensor node’ s resources and then, introduced share-able components that minimize the consequential resources needed and a resource planner that reserve these resources.

- Running multi-sequence applications in wireless sensor networks

  Amjed Majeed and Tanveer A Zia have built a middleware layer model where multiple applications can run on a single network on a time based sequence in a predefined order[13]. In this model, concurrency of applications in a sequenced order is achieved which results in prolonging the lifetime of sensor nodes. This is mainly achieved by having a switching architecture of network sub-sets between sleep and active modes based on which application is currently running. This model takes into consideration a dense wireless sensor network where multiple applications are running on different sets of sensor nodes alternatively executing the applications and saving the computational and memory resources.

2.4 Summary

In this chapter, we described the traditional WSNs and explained why we need the MA-WSNs. By creating MA-WSNs, we can reduce the administrated cost and deployment cost of WSNs. Since MA-WSNs is created as a part of global sensing infrastructure, many sensor networks applications can be widely, efficiently used. We discussed the requirements of an useful MA-WSN include of a flexible data collection protocol, an effective control protocol and an efficient routing protocol. Some of related infrastructures were also stated.

Chapter 3

TinyOS, Data Collection and Sensor Control Protocol in Wireless Sensor Network

This chapter represents TinyOS, an operating system for sensor networks, the data collection protocols and sensor control protocols, which are widely used in current wireless sensor networks.

3.1 TinyOS

TinyOS is a free and open source component-based operating system and platform targeting WSNs. TinyOS is an embedded operating system written in the nesC programming language, a dialect of the C language optimized for the memory limits of sensor node, as a set of cooperating tasks and processes. Started as a collaboration between the University of California, Berkeley in co-operation with Intel Research and Crossbow Technology, today, TinyOS is widely used in many WSNs.

TinyOS program are built out of software components, some of which present hardware abstractions. Components are connected to each other using interfaces. TinyOS provides default interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage. When a programmer want to build a TinyOS program, she has to build necessary components that TinyOS does not provide and wire them together.

TinyOS code is statically linked with program code, and compiled into a small binary, using a custom GNU toolchain. Associated utilities are provided to complete a development platform for working with TinyOS.

3.2 Relation of data collection and sensor control in wireless sensor network

3.2.1 Data collection protocol

Data collection protocol is the protocol that manage collecting the data generated in the network into a base station (or a sink node). The general approach used is to build one or more collection trees, each of which is rooted at a base station. When a sensor node has data which needs to be collected, it sends the data up the tree. Then the data will be forwaded to the sink node. Sensor nodes also forwards collection data that other nodes send to them.

Based on the architecture of WSNs, environment and the purpose of data collection, many data collection protocol[14][15][16][17][18][19] have been developed.

3.2.2 Sensor control protocol

Sensor control protocol is the protocol used to manage sensor nodes's behavior in wireless sensor network. The sink node use this protocol to send control message to sensor nodes to manage the sample data rate and sensed data type of sensor nodes. Since sensor nodes in MA-WSNs behave in different ways, this protocol is very important and affect the robustness of sensor networks.

3.2.3 Relation of data collection and sensor control in wireless sensor network

Wireless sensor network have been introduced and developed to monitor physical or environment conditions. Therefore, creating data related to the change of environment and transmitting these data to applications are tasks of WSNs. Because of this reason, data collection protocol and sensor control protocol are the most important protocols in WSNs. Sensor control protocol manage how the data are created at sensor nodes while data collection protocol control how these data can be sent to applications.

In tradittional WSNs, the only difference of sensor nodes is the location of them. Then, the sensor control protocol are not so important. But in the MA-WSNS, all of sensor nodes work in different ways hence there is an importance of how can we control the behavior of sensor nodes. A bad sensor control protocol lead to the large packet transmission overhead, data redundancy and packet interference of networks. It directly affects to the performance of data collection protocol.

Besides, since request in MA-WSNs is variety, inflexible data collection protocol can be the reason that make sensor nodes created unnecessary data as discussed in sections below. Hence, it take time of sensors to created data. Also, the risk of error will be increased which results in poor performance of sensor control protocol.

3.3 Data collection protocol in TinyOS

3.3.1 The Collection Tree Protocol

The Collection Tree Protocol (CTP)[20] is the default data collection protocol of TinyOS. It is a tree-based collection protocol. There are some special node in the WSN advertise themselves as tree roots. All of sensor nodes form a set of routing trees to these roots. CTP is address-free in that a node does not send a packet to a particular root; instead, it implicitly chooses a root by choosing a next hop. Nodes generate routes to roots using a routing gradient. CTP assumes that it has link quality estimates of some number of nearby neighbors. These provide an estimate of the number of transmissions it takes for the node to send a unicast whose acknowledgment is successfully received.

CTP uses expected transmissions (ETX) as its routing gradient. A root has an ETX of 0. The ETX of a node is the ETX of its parent plus the ETX of its link to its parent. This additive measure assumes that nodes use link-level retransmissions. A sensor node has one neighbor's information table that store the ETX of links to its neighbor. Given a choice of valid routes, CTP will choose the one with the lowest ETX value.

TinyOS uses the four-bit wireless link estimation protocol (4bitle) [21] to calculate the ETX used in CTP protocol. 4bitle uses two mechanisms to estimate the quality of a link: periodic LEEP packets and data packets. The protocol sends routing beacons as LEEP packets. This packets seed the neighbor table with bidirectional ETX values. It uses an algorithm similar to the trickle dissemination protocol[22] to adapt its beaconing rate. 4bitle augments the LEEP link estimates with data transmissions. Whenever the data path transmits a packet, it tells the link estimator the destination and whether it was successfully acknowledged or not. The estimator produces an ETX estimate every 5 such transmissions.

The CTP data frame format is described in figure 3.1

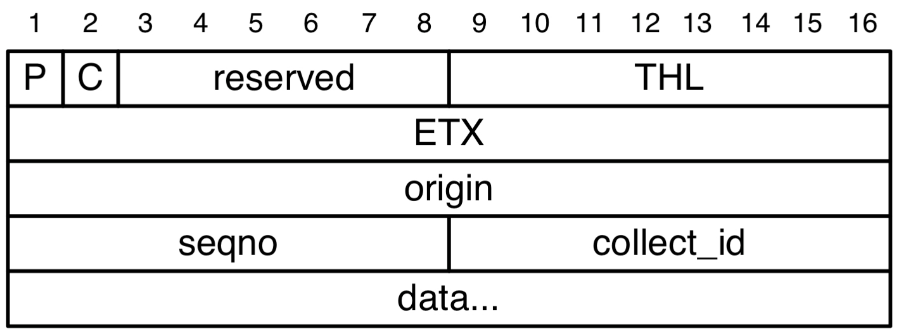


Figure 3.1: CTP data frame format

Field definitions are as follows:

- P: Routing pull. The P bit allows nodes to request routing information from other nodes. If a node received a packet with the P bit set, it will transmit a routing frame as soon as possible.
- C: Congestion notification. If a node drops a CTP data frame, it must set the C field on the next data frame.
- THL: Time has lived
- ETX: The ETX routing metric of the single-hop sender. When a node transmits a CTP data frame, it must put the ETX value of its route through the single-hop destination in the ETX field. If a node receives a packet with a lower gradient than its own, then it must schedule a routing frame in the near future.
- origin: The originating address of the packet
- seqno: Origin sequence number

- collect_id: Higher-level protocol identifier
- data: The data payload, of zero or more bytes

Together, the origin, seqno, collect_id and THL denote a unique packet instance within the network and used to detect the packet duplication.

The CTP routing frame formate is described in figure 3.2

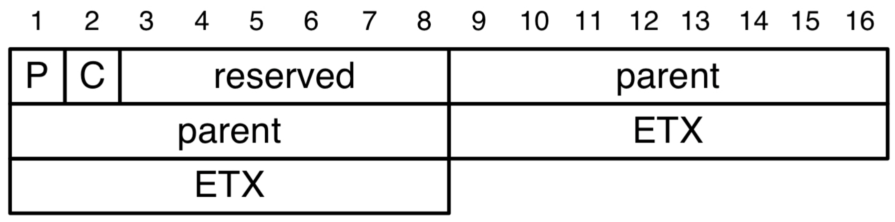


Figure 3.2: CTP routing frame format

Field description are as follows:

- P: Same as data frame
- C: Same as data frame but instead of CTP data frame, the C bit will be set on the next routing frame.
- parent: The node's current parent.
- ETX: The node's current routing metric value.

When a node hears a routing frame, it must update its routing table to reflect the address's new metric. If a node's ETX value changes significantly, then CTP should transmit a broadcast frame soon thereafter to notify other nodes, which might change their routes.

3.3.2 Problems

Since CTP is designed to support traditional WSNs, it lacks necessary functions and inflexible to used in MA-WSNs.

In the MA-WSNs, one sensor node can be requested by multiple applications. Obviously, different applications have different requests in terms of sample rate and data type. Because CTP protocol use unique data frame, to collect data for multiple applications, all of sensor nodes have to sense all of data types and transmit the unnecessary big data frame. Since sensor nodes almost use its energy on radion transmission, it increases the energy consumption and packet loss rate of the network.

TinyOS is built to use in the sensor networks where every nodes behave in the same way. Therefore, CTP only has the routing procol that is used to select path from sensor nodes to sink node. It does not have the reverse routing to select path from sink node to sensor nodes. In order to send control message to sensor nodes individually, this kind of routing is required so we can not use CTP in MA-WSNs.

3.4 Sensor control protocol in TinyOS

3.4.1 Dissemination

Since TinyOS only supports traditional WSNs, it does not have the protocol that allow sink node to send control message to sensor nodes individually. Instead, it uses dissemination protocol to send control message to all of sensor nodes.

Dissemination is a service for establishing eventual consistency on a shared variable. Every node in the network stores a copy of this variable. The dissemination service tells nodes when the value changes, and exchanges packets so it will reach eventual consistency across the network. At any given time, two nodes may disagree, but over time the number of disagreements will shrink and the network will converge on a single value.

There are three kinds of dissemination libraries in TinyOS: Drip, DIP, and DHV. All of these use Trickle[22] timer underneath for propagation in network. DIP treats every data item as a separate entity for dissemination, and thus provides fine grain control of when and how quickly you want data items disseminated. DIP and DHV treat them as a group, meaning dissemination control and parameters apply to all data items collectively.

In TinyOS, dissemination provides two interfaces: DisseminationValue and DisseminationUpdate. These interfaces are described as follows.

The DisseminationUpdate interface is used by producers while the DisseminationValue is used by consumers. The command DisseminationUpdate.change() is called each time the producer wants to disseminate a new value, passing this new value as a parameter. The event DisseminationValue.changed() is signalled each time the disseminated value is changed and the command DisseminationValue.get() allows to obtain this new value.

```
interface DisseminationUpdate<t> {
    command void change(t* newVal);
}

interface DisseminationValue<t> {
    command const t* get()
    event void changed();
```

Figure 3.3: Dissemination Protocol Interface

Commonly, dissemination protocols are used to reconfigure the sample rate of sensor nodes in WSNs.

3.4.2 Problems

As same as CTP protocol, dissemination protocol also inflexible and cannot be used in MA-WSNs.

Dissemination protocol send the control message to all of sensor nodes in network and force all of sensor nodes to work in the same way. But in MA-WSNs, letting sensor nodes to work in different ways based on applications request is better. Therefore, a sensor control protocol that can send control message to sensor nodes individually is necessary.

In MA-WSNs, a sensor node may be requested by multiple applications in different sample rates. By using dissemination protocol, this sensor node have to work in maximum sample rate that is requested. For example, a sensor node is requested by three applications. These applications request this sensor node have to send data with periods of 4, 6, and 7 second. Therefore, this sensor node have to transmit data for each second. It causes a large energy wastage, increase the packet transmission overhead, data redundancy, interference and packet loss in network.

3.5 Summary

In this chapter, we introduced TinyOS, a famous and widely used operating system in wireless sensor networks. We also introduced and discussed the relation and problems of current data collection protocol and sensor control protocol used in TinyOS. Since creating and transmitting data is the tasks of WSNs, these protocols are the most important protocols in WSNs. Because of the variety of request of applications in MA-WSNs, they become inflexible and lack of functions to support multiple application. They cause the large packet transmission overhead, data redundancy, packet interference, and energy wastage of wireless sensor network. For these reason, these protocols cannot be used and raise the needs of flexible and efficient data collection and sensor control protocols in MA-WSNs.

Chapter 4

Approach and System Design

This chapter represents the approach and the design of proposed system.

4.1 LEAF

This thesis proposes LEAF - a library used in TinyOS to support MA-WSNs. It is a set of protocols and management method of sensor nodes and sensor data that fulfill the requirement of MA-WSNs. LEAF uses a flexible data collection protocol and an efficient sensor control protocol to adapt with the variety of request from multiple applications. In addition, LEAF provides an application communication interface that not only supports local applications but also can serve the requests from external applications. The detail description of LEAF is introduced in sections bellow.

4.2 System overview

The overview of a MA-WSN that used LEAF is described in figure4.1 There are three main modules of LEAF: application communication module, sensor nodes control module and data collection module. The application communication module provides an communication interface between applications and sensor network. It receives requests from applications, processes it and passes the parameters to sensor nodes control module. The sensor nodes control module's tasks are transmiting the control message from the sink node to sensor nodes and controlling the sensed data type and sample rate of sensor nodes. The data collection module control the data transmission from sensor nodes to the sink node. Finally, a link estimation module is used to calculate the ETX (expected transmissions) that is used in routing protocols. Except the application communication protocol which is implemented only at the sink nodes, all modules are implemented at all of the sensor nodes as well as the sink nodes.

4.3 Applications communication

The Applications communication module is implemented at sink node and acts like a server that receives requests from applications and deliver sensor data to approriate application through the TCP/IP connection. By using the TCP/IP connection, it can receive not only the local request but also the external request.It has two sub module: the Request processing module and the Data transmission control module that is described as follows.

4.3.1 Processing Request

The Request processing module's task is to delimit each application to its specific set of relevant nodes. When a request from an application comes, this module is used to determine which sensors are required and the data type, the sample rate of these sensors. It also has to store the list of current applications that is used in data transmission control module. After that, it passes these parameters to the Sensor nodes control module.

4.3.2 Data transmission control

The Data transmission control module controls the data transmission between sensor network and applications. After receiving the data from Data collection module, based on the received data frame and the list of current applications that provided by the Request processing module, it can determine the address of approriate applications and send the data to these applications.

In MA-WSNs, the received data frame may contains the data from a sensor nodes to several applications. Hence, the Data transmission control module has to distinguish them and send the right data to the right application. Besides, one applications may require data from multiple sensor nodes so data have to be aggregated before sending them to applications.

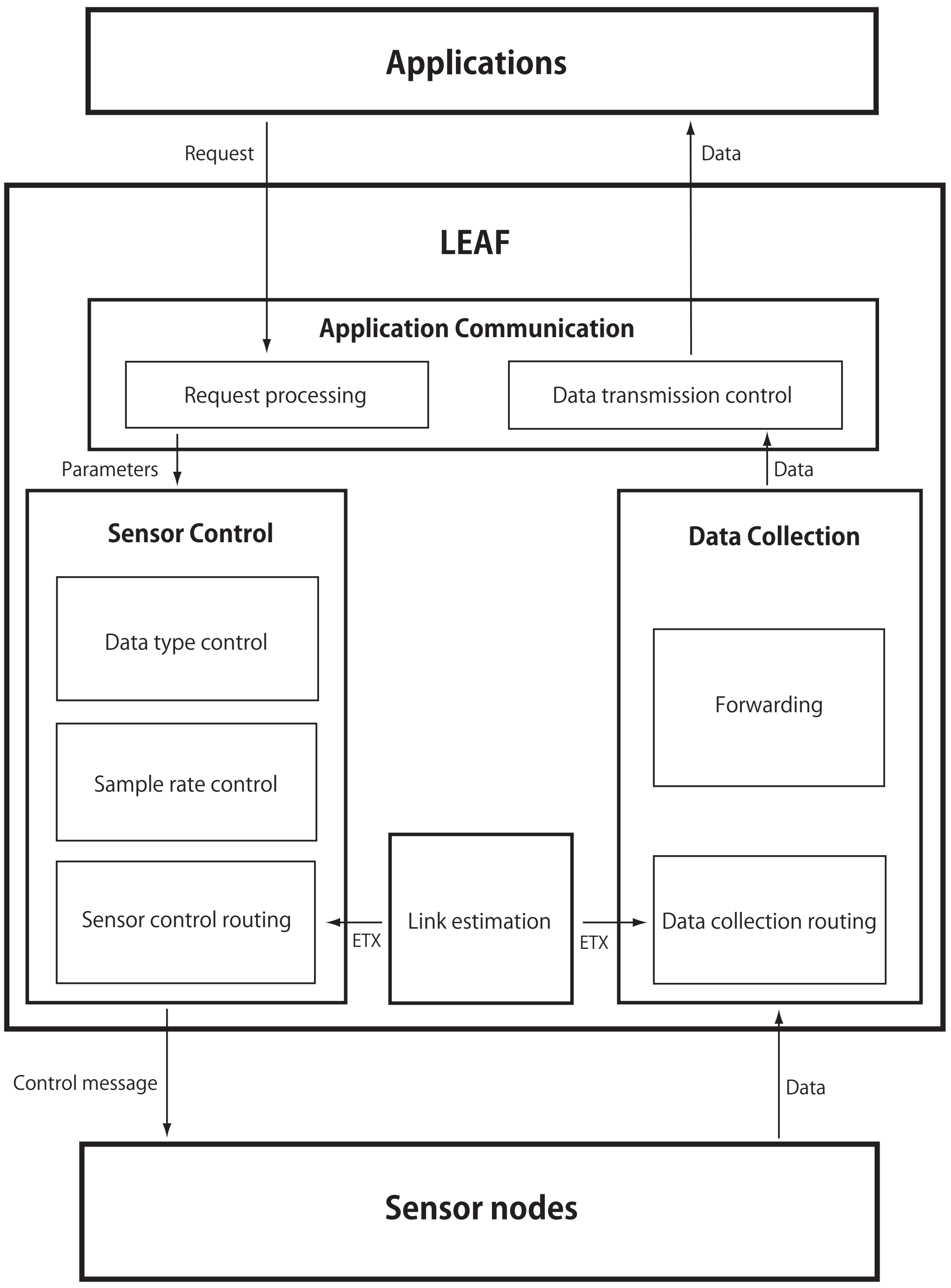


Figure 4.1: System overview

4.4 Sensor nodes control

Sensor nodes control module works in two parts. The first part is creating and sending the control message. This task is done in the sink node. After receiving the parameters from the Request processing module, the control messages which contains sensor id, application id, sample rate, data types will be created. Then, these control messages will be sent to related sensors through the path that is selected by the Sensor control routing module.The second part is done in the sensor nodes. After receiving the control message from the sink node, by using the Data type control module and the Sample rate control module, the sensor nodes's schedule will be updated.The Sensor nodes control also has a software acknowledgement mechanism to ensure the control message to be properly sent. After a predefined timeout, if the sink node did not receive the ACK from sensor nodes, it will send the control message again.

4.4.1 Data type control

As discussed in chapters above, in MA-WSNs, one sensor node is connected to several sensors then it can sense several types of data concurrently. But at one time, only a part of these types is required. Hence, we need to control this to prevent the data redundancy and packet loss in the network. That is the task of Data type control module.

The Data type control module is implemented and processed in sensor nodes. It stores a list of applications that currently work with sensor node and for each application, it stores a list of data types that are required by this application. Based on these lists, when the sensor node sense data (or measure the physical environment), it only sense for appropriate types.

4.4.2 Sample rate control

In MA-WSNs, a sensor node is requested by mutiple applications with various sample rate concurrently. The easiest way if forcing this sensor node to work with maximum of these sample rates. But it will result in large packet transmision overhead and packet interference in network. It also increase the packet loss rate. To solve this problem, we create the Sample rate control module that implemented and processed in the sensor nodes to manage the sample rate of them.

Similar to the Data type control module, the Sample rate control module stores a list of applications that currently work with sensor node too. A list of sample rate is also stored. Each of sample rate corresponding to an application in applications list. Using these lists, the Sample rate control module will schedule the sensor node to work in only necessary time. The sensor nodes do not sense data periodically but only sense data at the time that calculated.

4.4.3 Sensor control routing

The sensor control routing module is used to select path from the sink node to sensor nodes to send the control message. Basically, the path which is selected by this module is the reverse of the path which is selected by the Data collection routing module.

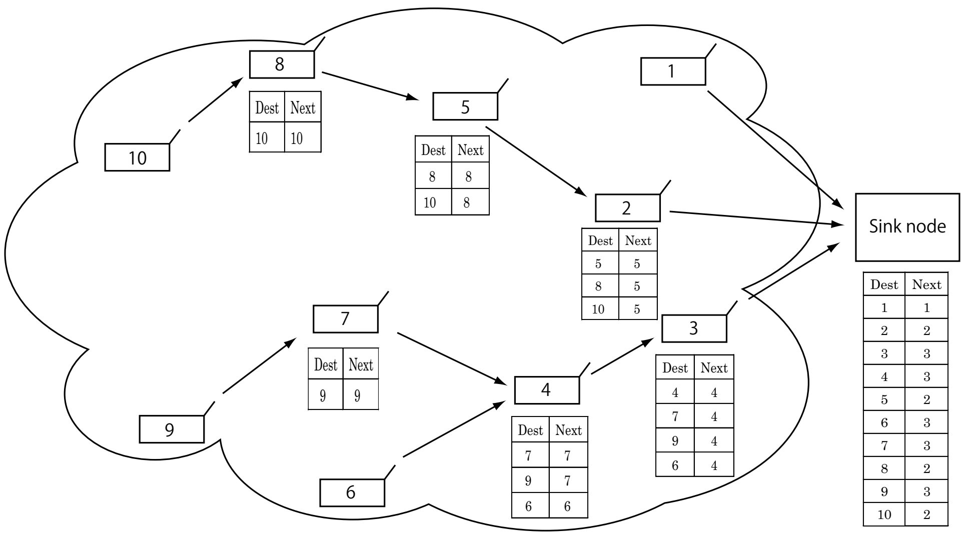
The sensor control routing module use the ETX which calculated by the Link estimation module to detect the best single-hop link from sensor node to its neighbors. The neighbor node that has best quality is called "parent node". When there are changes in network, the link estimator will recalculated the ETX of links and if the parent node changed, the sensor node has to send the routing message up the collection tree to the sink node. Because the routes from the sink node to all of nodes in the tree that has the root as this node have to be changed, the routing message has to contain all of these node's address. When a sensor node (also the sink node) receive the routing message, it will update its routing table and forward the message up the tree. Figure 4.2 illustrates the collection tree of WSN and how the routing tables are created.

Figure 4.3 shows what happen when the parent of a sensor node 7 changed from node 4 to node 5 and the routing message.

The Sensor control routing module also have a acknowledgement mechanism to ensure routing messages properly sent or not.

4.5 Data collection

LEAF uses a data collection protocol which is created based on the Collection Tree Protocol that was introduced in chapter 3 with some improvements. To support MA-WSNs, it uses the dynamic data frame format that contains application identifiers and the data will be various based on these identifiers.



Dest	Next
10	10

Dest	Next
8	8
10	8

Dest	Next
5	5
8	5
10	5

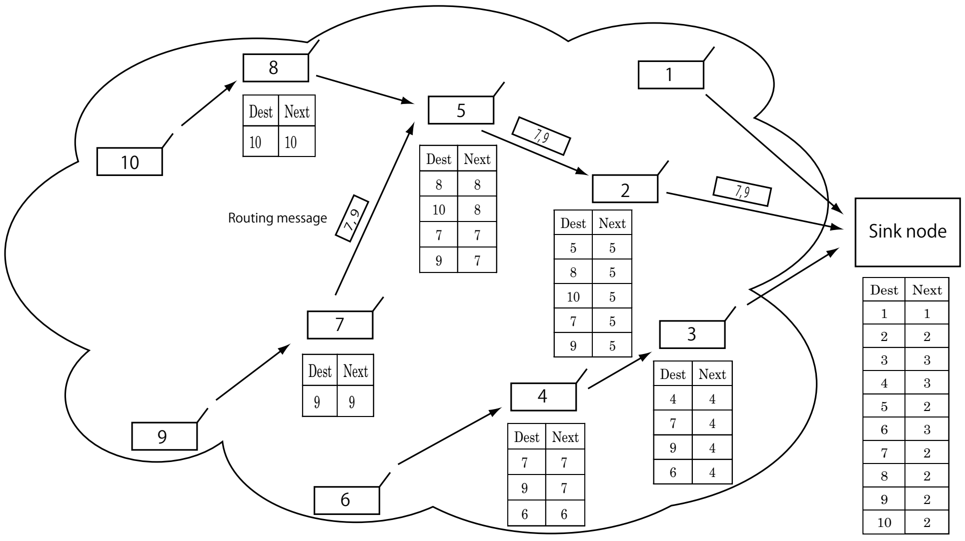
Dest	Next
9	9

Dest	Next
7	7
9	7
6	6

Dest	Next
4	4
7	4
9	4
6	4

Dest	Next
1	1
2	2
3	3
4	3
5	2
6	3
7	3
8	2
9	3
10	2

Figure 4.2: Sensor control routing scheme



Dest	Next
10	10

Dest	Next
8	8
10	8
7	7
9	7

Dest	Next
5	5
8	5
10	5
7	5
9	5

Dest	Next
9	9

Dest	Next
7	7
9	7
6	6

Dest	Next
4	4
7	4
9	4
6	4

Dest	Next
1	1
2	2
3	3
4	3
5	2
6	3
7	2
8	2
9	2
10	2

Figure 4.3: Routing message transmission and routing table update

4.5.1 Data collection routing

Similar to the Sensor control routing module, the Data collection routing module also use the ETX that calculated by the Link Estimation module to decide the parent of sensor nodes. In addition, it also use the hardware information on computing route. These informations come from MAC layer such as RSSI, LQI of data packet.

Different than the Sensor control routing, the Data collection routing only compute the single-hop route. It keeps track of the path ETX values of a subset of the nodes maintained by the link estimation table. The path ETX is therefore the sum of link ETX values along the entire route. Routing mechanism will pick the path that has minimum path ETX.

4.5.2 Data forwarding

The main repsonsibility of the Data forwarding module is transmistting data packets to the next hop and retransmitting when necessary. It also passing the acknowledgement based information to the Link Estimation module to recompute the link quality.

Data forwarding module has a congestion control mechanism to decide when to transmit packets to the next hop. It only send data if the next hop is not in the busy mode.

Detecting routing inconsistencies to inform the routing engine and detecting single-hop transmission duplicates caused by lost acknowledgement are also tasks of this module.

4.6 Link estimation

The Link estimation module is used to calculate the ETX of the single-hop link between sensor nodes. It uses two mechanisms to estimate the quality of a link. The first is periodic routing beacons. These beacons packet seed the neighbor table with bidirectional ETX values. The second is using the data packet transmission. When a data packet was transmitted, the Data forwarding module will notice the Link estimation module that the data packet was successfully acknoledged or not. Based on these information, the Link estimator will recompute the ETX of the link.

4.7 Disscussion

LEAF, which contains a set of modules that was described above is robust, efficient and flexible to use in MA-WSNs.

Firstly, it has an interface that can receive and process request from multiple applications. By acting like a server, this interface not only can serve the local requests but also can response to the external requests, such as the requests come from the mobile applications.

LEAF uses a sensor control module, which contain the flexible protocols that can manage and schedule the sensor nodes as well as sensor data efficiently. The sensor nodes only have to do the necessary task and send the necessary data, which results in the decrease of the packet transmission overhead, the data redundancy, the packet interference and the enery consumption of the sensor networks. Hence, it successfully adapt to the variety and the dynamic of MA-WSNs.

4.8 Summary

This chapter introduced and discribed LEAF, a library that support MA-WSNs and its modules. We also show that how the system that uses LEAF works. By the disscusion about LEAF, we demonstrated that LEAF contains robust, high efficiency, high flexibilily management methods and totally can be used to create the MA-WSNs.

Chapter 5

Implementation of LEAF

This chapter describes the implementation of LEAF. Firstly, we will introduce the implementation platform of LEAF. Then, all of modules will be described respectively. Finally, the usage of LEAF and summary of chapter will be stated.

5.1 Implementation platform

The most widely used wireless sensor node is mote. There are many type of mote, such as Iris, Mica2, Micaz, Telos, etc. Also, TinyOS can run on mote. In this thesis, we implemented LEAF on Iris mote (figure 5.1(a)), a successor of Micaz mote, which is used in many WSNs research. The specification of Iris mote is shown in table 5.1 below.

Table 5.1: Specification of Iris mote

CPU	Atmega1281
Clock Speed	7.37 MHz
Flash Memory	128kB
RAM	8kB
Radio Chip	RF230
Frequency Band	2,400-2,480MHz
TX Data Rate	250kbps

Iris mote can work with many sensor boards such as MTS300/310, MTS400/420, MDA100, MDA300, etc. LEAF is implemeted with MTS400 sensor board (figure 5.1(b)). The intergrated sensors of MTS400 sensor board are shown in table 5.2.

(a) Iris mote senor node

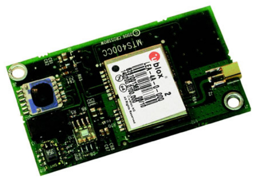

(b) MTS400 sensor board

Figure 5.1: Iris mote and MTS400 sensor board

Table 5.2: Intergrated sensors of MTS400

Humidity and Temperature	Sensirion SHT11
Barometric Pressure and Temperature	Intersema MS5534
Light	Taos TSL2550
2-Axis Accelerometer	Analog Devices ADXL202JE

LEAF is implemented by Java and Network Embedded Systems C (NesC) programming language.

5.2 Applications communication

This is the only module that implemented by Java in LEAF. It builds a server to communicate with applications through TCP/IP connections. This module interacts with the sensor network through a sink node. The parameters and the sensor data are passed through a serial connection.

The structure of request messages from applications is shown in figure 5.2.

We use the Bitwise operation to mask what sensors to request and what types of data are need. After receiving the request messages from applica-

```
typedef nx_struct request_message {
        nx_uint64_t sensor_list;
        nx_uint8_t rate;
        nx_type_t type;
} request_message_t;
```

Figure 5.2: Request Message Structure

tions, the computer send serial messages to the sink node (this is the control message but send via serial port so we call it serial message). The structure of serial message is illustrated as in figure 5.3.

```
typedef nx_struct serial_message {
        nx_am_addr_t dest;
        nx_uint8_t app_id;
        nx_uint8_t rate;
        nx_type_t type;
} serial_message_t;
```

Figure 5.3: Serial Message Structure

For external requests, there is a need of a global infrastructure that can provide sensor identifier to application developers such as the infrastructure named SensingCloud[23], which is an open and global sensor network using distributed aggregation mechanism built by Namatame et al.

5.3 Sensor control module

This module is implemented at all of the nodes in network by NesC programming language.

After receiving the request from an application, the control message will be created and sent to relevant sensor nodes through the path that selected by sensor control routing module. The structure of control message is described in figure 5.4. All of parameters are received from the Request processing module.

```
typedef nx_struct control_message {
        nx_uint16_t seqno;
        nx_am_addr_t dest;
        nx_uint8_t app_id;
        nx_uint8_t rate;
        nx_type_t type;
        nx_uint8_t retries;
} control_message_t;
```

Figure 5.4: Control Message Structure

5.3.1 Data type control

To control the data types of sensor nodes, the Data type control module maintain a table that store informations of applications that currently work with them. The entry of this table has structure that described in figure 5.5

```
typedef nx_struct app_data_type {
        nx_uint8_t      app_id;
        nx_type_t       type;
} app_data_type_t;
```

Figure 5.5: Data Type Control Table Entry Structure

When the Samle rate control module raise a signal to notice that the sensor node has to sense data, the Data type control module has to process this signal. Based on the data type control list and the list of applications come from the Sample rate control module, it will decide what types of data to sense and then, create a data message to transmit to the sink node.

The Data type control module provides the LeafDataTypeControl interface that described below.

The **add** command and **delete** are used by Sensor control module when sensor node receives the control message.

Since we implemented LEAF on Iris mote with mts400 sensor board, we have the sensor data set that illustrated in figure 5.7.

```
interface LeafDataTypeControl {
    command error_t add(uint8_t app_id, type_t type);
    command error_t delete(uint8_t app_id);
    command type_t getDataType (uint8_t app_id);
}
```

Figure 5.6: LeafDataTypeControl Interface

```
typedef struct {
    uint16_t temp;
    uint16_t humidity;
    uint16_t accelx;
    uint16_t accely;
    uint16_t pressure;
    uint16_t vlight;
    uint16_t ilight;
} SENSOR_DATA_SET;
```

Figure 5.7: Sensor Data Set Structure

5.3.2 Sample rate control

The Sample rate control module stores a table of applications and the sample rate of these applications. The sample rate is stored as the length of time between consecutive sensings with the unit of second. For example, an application require data at 0.5Hz means that there are 2 seconds between consecutive sensings, so its sample rate will be 2. The structure of the sample rate table entry is illustrated in figure 5.8.

Table 5.3 represents an example of the Sample rate control table.

Based on this table, the Sample rate control module will calculate the time that sensor node has to measure the physical quantity. Figure 5.9 illustrates how the Sample rate control module work with the example above.

Start with $t = 0$ when sensor node senses all of types of data that required by all of applications, the next sensing time is calculated based

```
typedef nx_struct {
        nx_uint8_t id;
        nx_uint8_t rate;
} app_sample_rate_t;
```

Figure 5.8: Sample Rate Control Table Entry

Table 5.3: Example of sample rate control table

Application_id	**Sample rate**
1	2
2	3
3	4

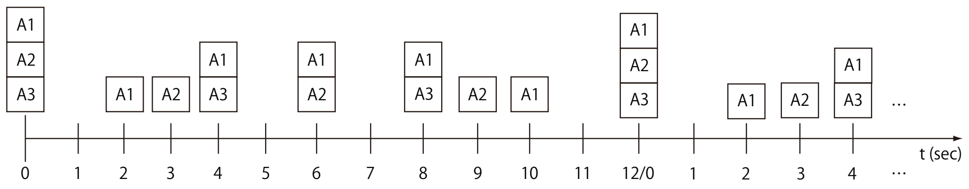


Figure 5.9: How sample rate control module work

on the current time and the informations provided by Sample rate control table. Notice that the sensing behavior is cyclical with the period equals to the Least Common Multiple of sample rates. In this example, we have the Least Common Multiple $LCM(2, 3, 4) = 12$ so the schedule will be reseted at $t = 12(s)$.

The LeafSampleRateControl interface provided by this module has the signature as shown in figure 5.10.

Similar to the Data type control module, the **add** and **delete** commands are used when sensor nodes received the control message. The fire event is signaled when the time to sense and transmit data come. The Data Type Control module will catch this event and create the data message to send to the sink node.

```
interface LeafSampleRateControl {
    command error_t add(uint8_t app_id, uint8_t sample_rate);
    command error_t delete(uint8_t app_id);
    event void fired(app_list_t app_list, error_t error);
}
```

Figure 5.10: Sample Rate Control Table Entry

5.3.3 Sensor control routing

At every sensor node, there is a Sensor control routing table, which stores the path to all of its successor nodes in the data collection tree. The fig 5.11 illustrates the entry of the sensor control routing table. Each entry has two fields, the destination node and the next node. When a control message arrives, sensor node will search for the next node to forward this message based on the destination field's value in the message. If no entry match, the message will be discarded.

```
typedef struct routing_table_entry{
                am_addr_t dest;
                am_addr_t next;
} routing_table_entry_t;
```

Figure 5.11: Sensor Control Routing Table Entry

The sensor control routing module provide the LeafSensorControlRouting interface, which has the signature as shown in the figure 5.12.

The **nextHop** command provides the address of the next node which parameter is the destination node. The command **hasRoute** determines the route to the **dest** exist or not. The command **update** is used to update or insert the route information to the Sensor control routing table. When the parent of a sensor node changes, it will send the information of new parent to the sink node. On the path from this node to the sink node, all of the sensor nodes that the message crosses will use this command to update their routing table. The update command is also used when the data

```
interface LeafSensorControlRouting {
        command am_addr_t nextHop(am_addr_t dest);
        command bool hasRoute(am_addr_t dest);
        command void update(am_addr_t dest, am_addr_t next);
}
```

Figure 5.12: LeafSensorControlRouting interface

message crosses it.

5.3.4 Sensor control forwarding

This module is used to send the control message to the relevant sensor nodes. It uses the route information that is provided by the Sensor control routing module to send the message. Figure 5.13 represents two main interfaces that are provided by the Sensor control forwarding module, **Send** and **Receive** interface. The **Send** inteface is used by sink node when it wants to send the control message to sensor node while the Receive.receive event is signaled when the control message arrived at relevant nodes.

```
module LeafSensorControlForwardingP {
        provides {
                interface Send;
                interface Receive;
        }
}
```

Figure 5.13: LeafSensorControlForwarding module

5.4 Architecture of LEAF

The overall wiring architecture of LEAF is illustrated in figure 5.14. The thick squares indicate the main components we implemented: LeafSensorControlP, LeafSampleRateControlP, LeafDataTypeControlP, LeafSensorControlForwardingP, and LeafSensorControlRoutingP. The thin squares indicate the generic components used by our system, and names beside the arrows represent the interfaces used to connect components.

Beside the main modules of LEAF, we uses the CTP data collection protocol to collect data. CTP provides a component called CollectionC and we modified it to provide the OnPathReceive interface that used in Sensor Control Routing module. The Link Estimator used in LEAF is the default of TinyOS - 4bitle protocol. After that, we use accompanied interfaces with mts400 sensor board to provide data.

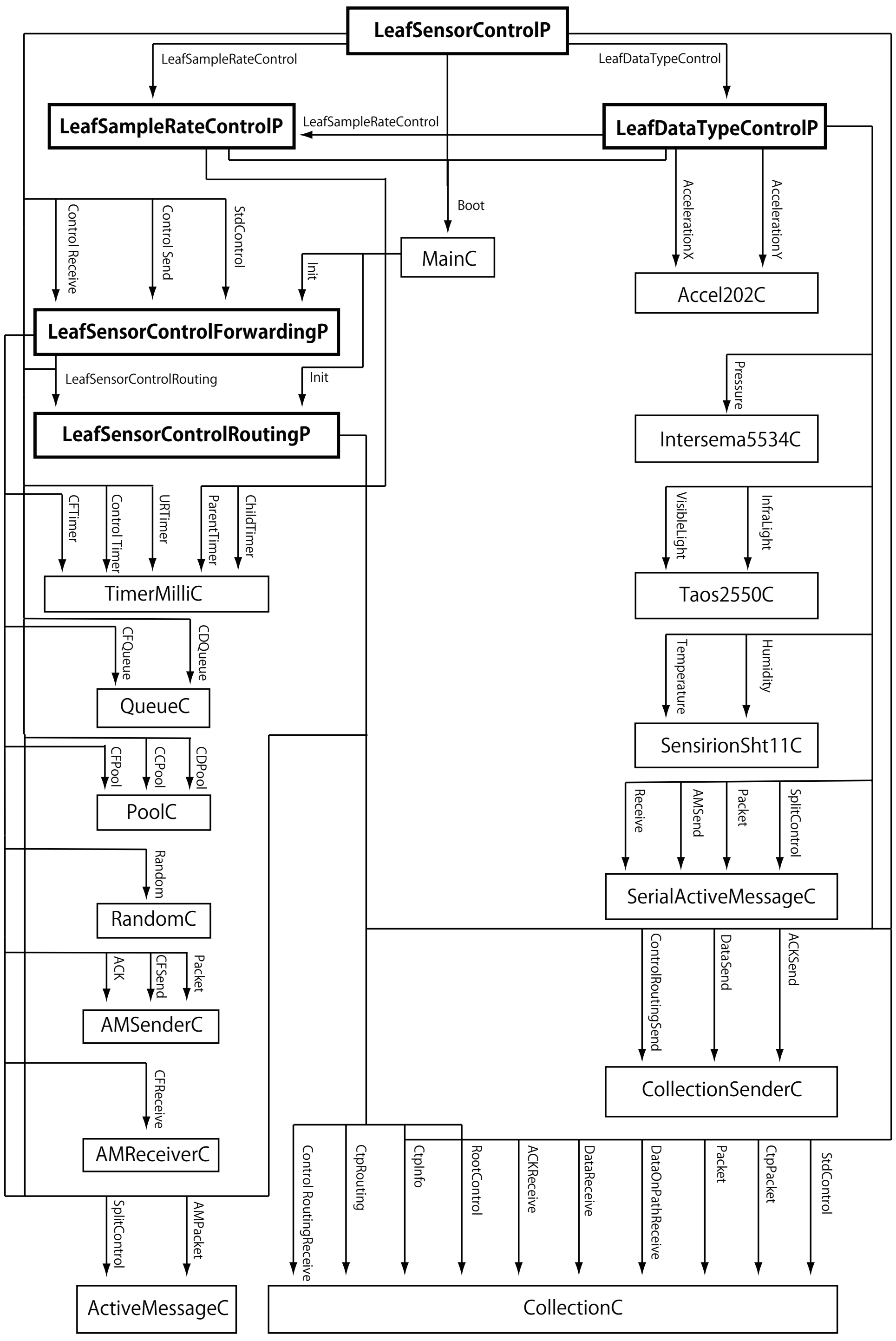


Figure 5.14: Overall wiring of LEAF

5.5 Summary

In this chapter, we represented the implementation of LEAF. We implemented LEAF on Iris mote in TinyOS environment by NesC and Java programming language. The main module of LEAF are Sensor Control, Sensor Control Routing, Sensor Control Forwaring, Data Type Control, Sample Rate Control and Application Communication module. Beside these, we use CTP data collection protocol, 4bitle Link Estimator Protocol and accompanied interfaces that provided by TinyOS.

Chapter 6

Evaluation

This chapter presents the evaluation of LEAF. We first explain the evaluation items and then we describe the evaluation methodology, results and the discussions.

6.1 Evaluation Items

This section describes the evaluation items and the reason we choose them.

The goal of LEAF is to provide a robust library that support MA-WSNs. It is useful only when it can efficiently manage the sensor network and let applications run smoothly. In order for LEAF to be used as a solution for management problem in MA-WSNs, it has to fulfill the requirements of MA-WSNs. Hence, we need to evaluate all of factors affect the validity of a method to solve the management problem in MA-WSNs.

- Packet Delivery Ratio
  The ultimate goal of a sensor network is to provide the informations about the physical environment to the applications. Properly transmit sensor data to applications is the minimum requirement of a good management method, especially, in MA-WSNs, where there are many applications with the variety of request. That is the reason why LEAP is evaluated in perspective of packet delivery ratio. Since the packet delivery ratio signifies the performance of sensor networks, it is desired to be as high as possible.

- Packet Retransmission
  A good management method not only has a good communication with

applications, but also has a good control policy inside the network. A bad control policy can be the reason of packet interference, packet loss in network. Hence, packets are retransmitted many times. Using the packet retransmission as a evaluation item, we can see how good LEAF's control policy is. Despite the fact that packet retransmission is essential to increase the packet delivery ratio of network, it causes high overhead not only at the sender nodes, but also at its neighbor nodes. Therefore, it is preferable to have high delivery ratio and the small number of retransmission at the same time.

- Packet Delay Variation
  The packet delay variation is used to test the stability of the network that uses LEAF as a management method. If the packet delay variation is large, it means that there are problems in the data transmission of the network. We also can see the ability to provide the sensor data at accurate sample rate which become very important in some case such as pattern recognition applications of LEAF. The packet delay variation is desired to be as low as possible.

- Energy Consumption
  Since the most notable characteristic of wireless sensor nodes is having severely limited energy source, the energy consumption is one of the most important factors that affect to the perfomance of sensor networks. A good management method has to have a good sensor nodes scheduling that efficienty uses the energy for sensing and transmitting data. By this reason, we use the energy consumption to evaluate LEAF. Of course, we expect the energy consumption to be as low as possible.

- Storage Usage
  Typically, extending the message queue size will increase the packet delivery ratio and decrease the packet retransmission of sensor network. However, in common, the wirless sensor nodes are equipped with small storage. Hence, it is important for a management method to have small storage usage then the remain memory can be utilized to increase the message queue size.

6.2 Real World Deployment

This section describe the real world deployment evaluation of LEAF.

6.2.1 Evaluation methodology

Evaluation Environment

The evaluation environment is shown in table 6.1

Table 6.1: Evaluation environment

Operating Systems	Mac OS X 10.6.8
TinyOS version	2.1.1
nesC version	1.3.3
AVR-GCC version	4.1.2
Java version	Open JDK 1.7.0
Platform	Iris mote (XM2110J)
CPU	Atmega1281
Radio Chip	RF230

We used latest verison TinyOS, nesC, AVR-GCC and Java JDK at Octorber 2011. LEAF is implemented on Iris mote, wich has Atmega1281 as CPU and use RF230 as radio transmission.

Experiments Setup

In our evaluation, we use fifty six Iris mote to create a MA-WSNs, one for sink node and fifty five sensor nodes. The sink node is connected to a personal computer via MIB520, which is a USB gateway that provides personal computers the connectivity to Iris, Micaz and Mica2 mote. Sensor nodes are attached with MTS400, a sensor board provided by Crossbow company, which can acquire temperature, acceleration, barometric pressure, ambient light and humidity.

We deploy a MA-WSNs at first floor of Delta building of Keio University, Shonan Fujisawa Campus. It is a two floors building contains research laboratories. Sensor nodes are deployed as show in figure 6.1.

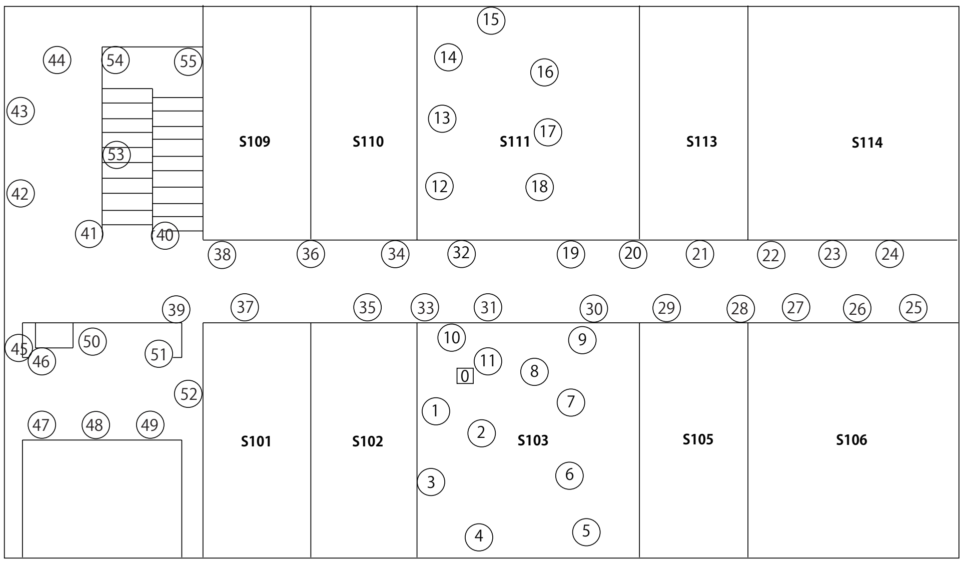


Figure 6.1: Sensor deployment

The evaluation will be held in many test case of applications to consider the ability of LEAF to adapt various kind of applications. The evaluation result of LEAF will be compare with the combination of CTP collection protocol and Drip, a widely used dissemination protocol in area of WSNs.

How to measure evaluation items

- Packet Delivery Ratio

  The sequence number is added into every transmission packet then when receiver receive a packet, it can compute the number of received packet and the packet that a sensor nodes sent.

  To observe the effect of the Sample Rate Control module and the Data Type Control module to the packet reception rate, the evaluation is held in 5 test cases, each have duration of 2 hours.

  – Test case 1

    In the first test case, we have a multi-application wireless sensor network with 3 applications request same data type and rate. Summary of applications in test case 1 is shown in table 6.2. This test case is used to measured the packet delivery ratio of LEAF in almost current WSNs, in which all of sensor nodes behave in the same way.

Table 6.2: Summary of applications used in test case 1

Number of applications	3
Required sensor	All 55 motes
Data Type	Temperature, Humidity
Packet interval	10 seconds

– Test case 2
  In this test case, there are 3 applications concurrently work in a WSNs. The information of these applications are shown in table 6.3. All of applications required same data types but in different sample rate. The results of this test case will show the effect of the Sample Rate Control module to the performance of network.

Table 6.3: Summary of applications used in test case 2

Number of applications	3
Required sensor	All 55 motes
Data Type	Temperature, Humidity
Packet interval	Application 1: 15 seconds Application 2: 35 seconds Application 3: 60 seconds

– Test case 3
  There are 3 applications in this test case. Table 6.4 show the summary of these applications. In this test case, the required packet intervals of all of applications are divisible by the minimum required packet intervals. Hence, the sensor nodes have to work with this sample rate. All of applications have different required data type than the others. Since the Sample Rate Control module does not significantly affect to the performance of network, this test case is used to evaluate the effective of the Data Type Control module.

– Test case 4
  This test case also have 3 concurrent applications. Their required data type and sample rate are different to the others. Table 6.5

Table 6.4: Summary of applications used in test case 3

Number of applications	3
Required sensor	All 55 motes
Data Type	Application 1: Temperature Application 2: Temperature, Humidity, Acceleration X Application 3: Temperature, Humidity, Acceleration X, Acceleration Y
Packet interval	Application 1: 20 seconds Application 2: 40 seconds Application 3: 60 seconds

show the information about these applications. This test case is used to evaluate the effect of combination of the Sample Rate Control module and the Data Type Control module to the performance of sensor network.

Table 6.5: Summary of applications used in test case 4

Number of applications	3
Required sensor	All 55 motes
Data Type	Application 1: Temperature Application 2: Temperature, Humidity, Acceleration X Application 3: Temperature, Humidity, Acceleration X, Acceleration Y
Packet interval	Application 1: 15 seconds Application 2: 35 seconds Application 3: 60 seconds

– Test case 5

Finally, we evaluate the packet delivery ratio of LEAF in one common case of MA-WSNs. There are 5 applications in this test

case. The description of these applications is described below.
The first application is assumed as a temperature control application of building. Its summary is shown in table 6.6.

Table 6.6: Summary of the first applications used in test case 5

Number of sensors	19
Required sensor	1, 5, 9, 13, 17, 20, 22, 23, 25, 28, 33, 36, 41, 44, 45, 47, 50, 52, 55
Data Type	Temperature
Packet interval	25 seconds

The second application is assumed as a temperature, humidity control application of building. We also assumed that sensor nodes used in this application are attached to actuator equipment like fan, etc, so the acceleration data is necessary. Summary of this application is shown in table 6.7 below.

Table 6.7: Summary of the second applications used in test case 5

Number of sensors	24
Required sensor	2, 4, 7, 8, 11, 13, 17, 20, 21, 24, 27, 28, 31, 34, 37, 40, 42, 44, 45, 46, 48, 51, 52, 55
Data Type	Temperature, Humidity, Acceleration X
Packet interval	25 seconds

The third application is assumed as a local environment monitoring application of the S103 room. Its information are shown in table 6.8.
Similar to the third application, the fourth application is assumed as a local environment monitoring application of the S111 room and its summary is described in table 6.9.
Last application is assumed as a corridor control application. Table 6.10 show the information of this application.

Table 6.8: Summary of the third applications used in test case 5

Number of sensors	1
Required sensor	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Data Type	Temperature, Humidity, Acceleration X
Packet interval	15 seconds

Table 6.9: Summary of the fourth applications used in test case 5

Number of sensors	7
Required sensor	12, 13, 14, 15, 16, 17, 18
Data Type	Temperature, Humidity, Acceleration X, Acceleration Y, Barometric Pressure
Packet interval	60 seconds

Table 6.10: Summary of the fifth applications used in test case 5

Number of sensors	23
Required sensor	19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41
Data Type	Temperature, Humidity, Acceleration X, Acceleration Y
Packet interval	40 seconds

Application types in test case 5 are common application types of WSNs and widely used in buildings and offices so it is reasonable to evaluate LEAF's performance in this case.

- Number of Packet Retransmission
  We measure the number of packet retransmission of both of control message and data message. For control message, we modified the LeafSensorControlForwarding module to add retransmission field to each control message packet. When a node retry to send a control message,

this field's value will be increased. After received the control message, a sensor node will transmited the ACK packet, which has retransmission field's value equal to correspond receive message value, back to root node. By this way, we can compute the number of retransmission of control message.
Similarly, for data message, we modified CTP to add retransmission field to CTP packet header and increase its value whenever a sensor node retry to transmit packet.
We evaluate the number of packet retransmission in three test cases, correspond to test case 3, 4, 5 in the evaluation of packet delivery ratio.

- Packet Delay Variation
  The packet delay variation is used to determine the ability of LEAF in perspective of transmiting data at right frequency that required by applications. The packet delay variation is computed at applications and compared to the applications's desire sample rate. When a packet came, the applications have to store the timestamp when they received packet. Finally, based on this list, we can compute the packet delay variation.
  We compute the packet delay variation in five test cases as same as test cases in packet delivery ratio evaluation.

- Energy Consumption
  We use rechargeable battery to measure energy consumption of sensor networks. When a node transmit data to sink node, it also include the voltage value of its battery. Because the voltage of batteries have large jitter when measure at sensor nodes, we use the average of all sensor nodes to minimize the errors.
  The energy consumption is evaluated in the test case 5 with duration of 11 hours.

- Storage Usage
  Since the sink node receive many data packet, for a fair comparison in perspective of packet delivery ratio and number of packet retransmission, at the sink node, we use as large as possible message queue size for transfering received data packets to the computer. However, in the sensor node, we use same size of forwarding queue. The storage

usage result is taken from the output of NesC compiler.

6.2.2 Evaluation Results and Discussion

Packet delivery ratio

Through the evaluation, we confirmed that LEAF outperforms the combination of CTP and Drip in perspective of packet delivery ratio. According to table 6.11, we can see that the packet delivery ratio of LEAF is approximately 99% while the best of CTP+Drip is only 78.6%, the average packet delivery ratio of CTP+Drip is 62%.

Table 6.11: Packet Delivery Ratio

Case	LEAF	Ctp+Drip
1	0.997	0.641
2	0.980	0.587
3	0.999	0.589
4	0.993	0.500
5	1.000	0.786

The packet delivery ratio in five test cases are shown in figure 6.2, 6.3, 6.4, 6.5, 6.6 respectively. In each figure, x-axis indicates the sensor node's ID, and y-axis represents the delivery ratio. The red x indicates the delivery ratio of LEAF which the blue one indicates the delivery ratio of Ctp+Drip combination. The horizontal lines indicate the average of all nodes. The black one shows the result of LEAF while the green one show Ctp+Drip's result. As shown in these figures, we can see that LEAF always has higher packet delivery ratio in all test case at all sensor nodes.

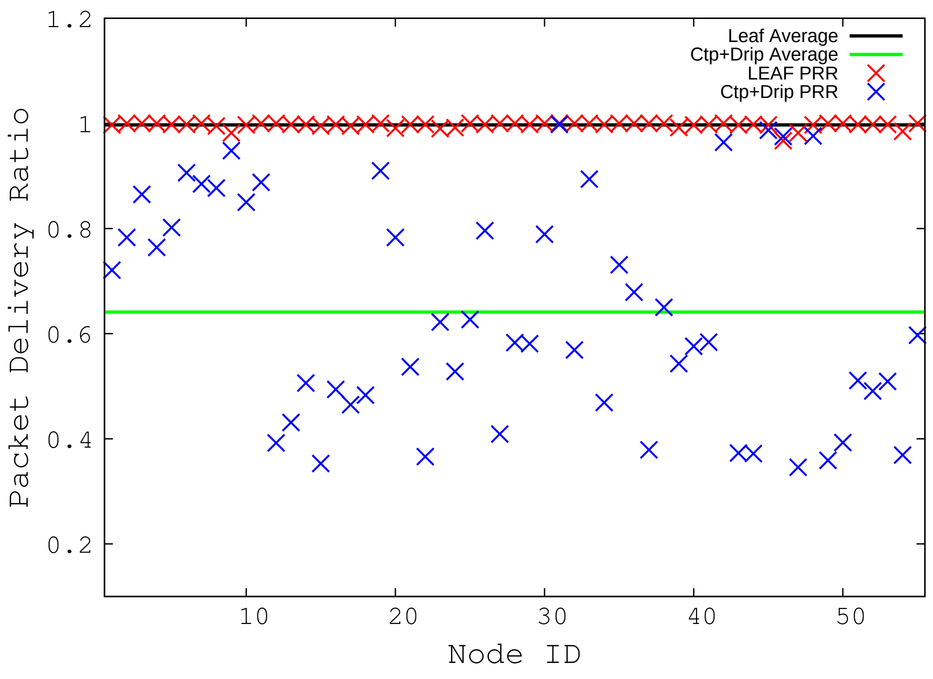


Figure 6.2: Packet delivery ratio in the first test case

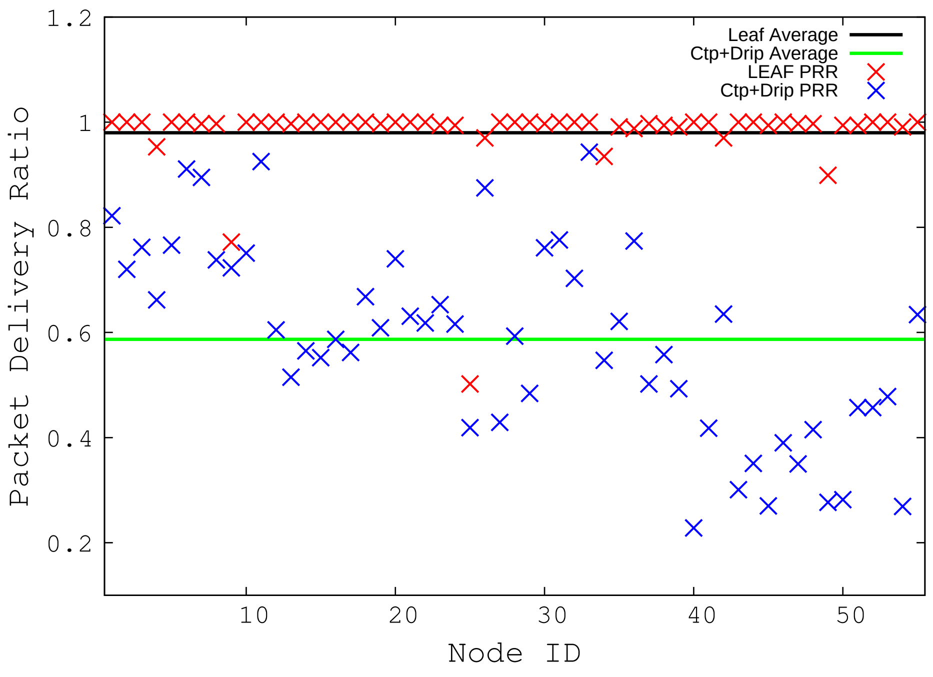


Figure 6.3: Packet delivery ratio in the second test case

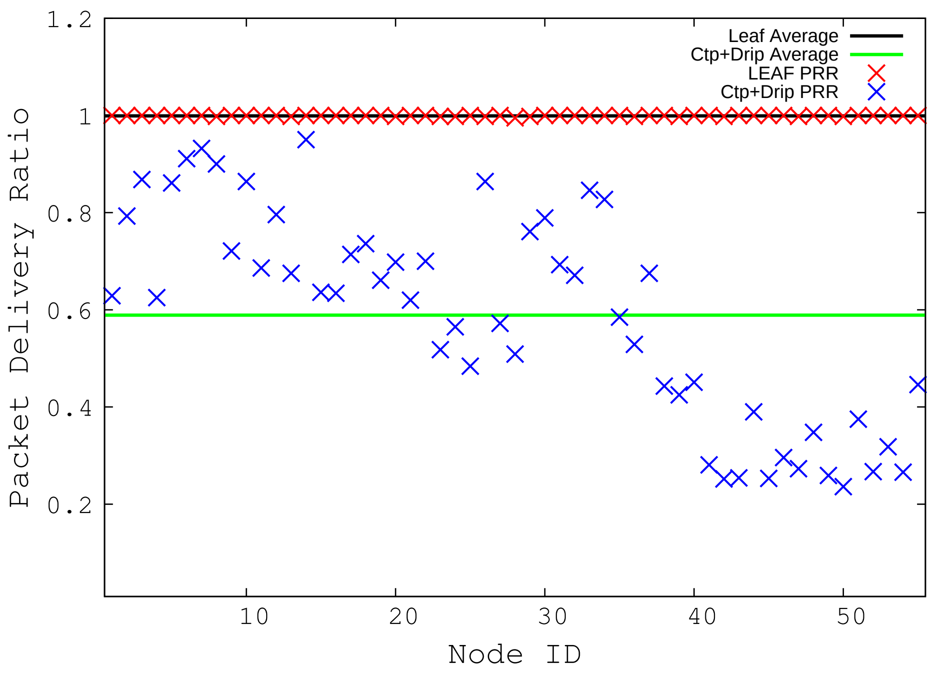


Figure 6.4: Packet delivery ratio in the third test case

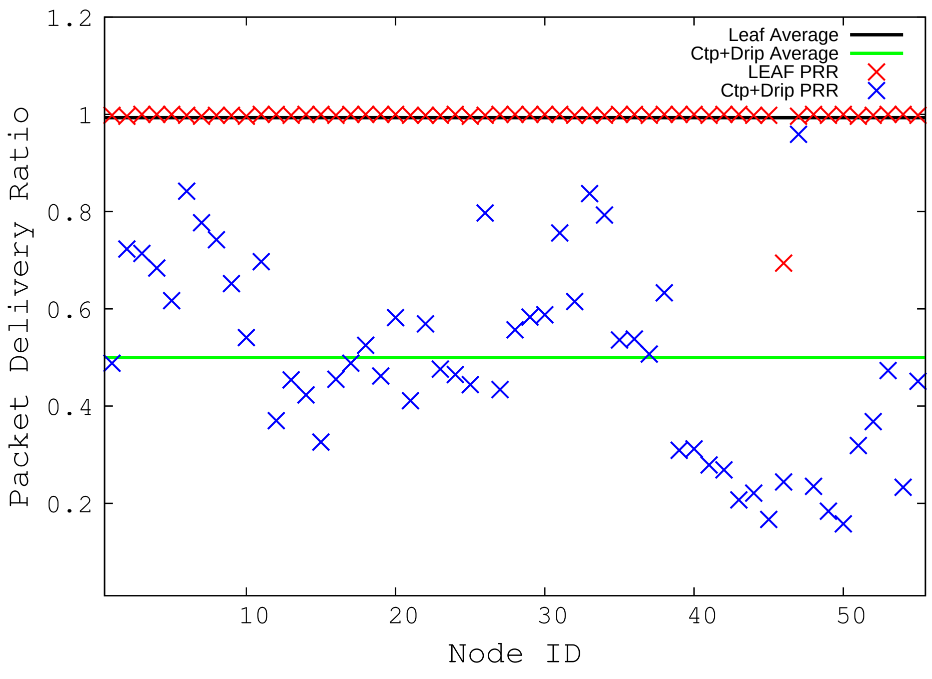


Figure 6.5: Packet delivery ratio in the fourth test case

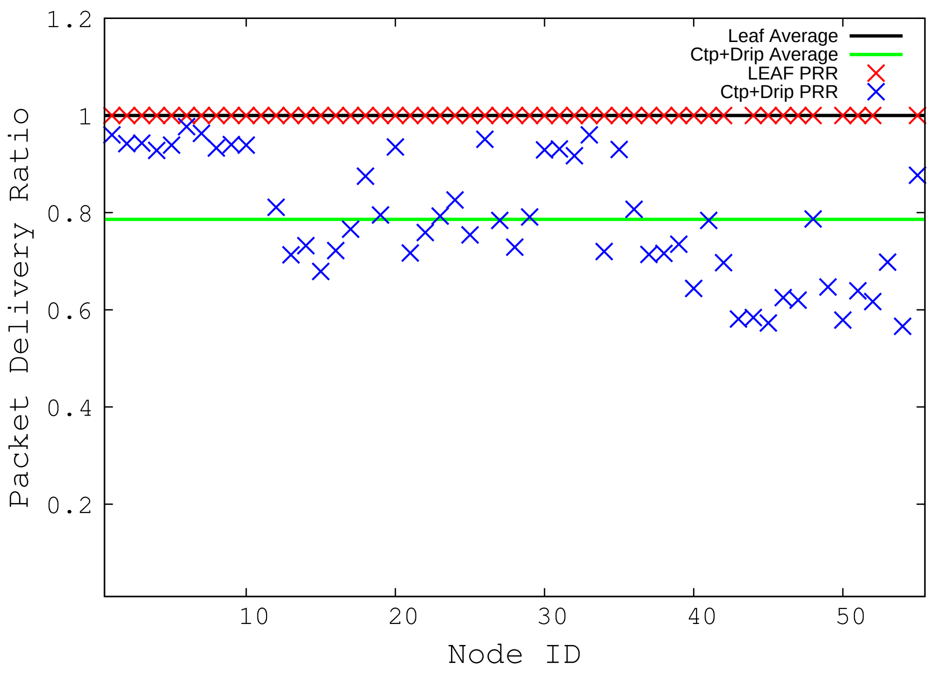


Figure 6.6: Packet reception rate in the fifth test case

While LEAF also uses CTP as its data collection protocol, it always achieve the higher packet delivery ratio because of three reasons. Firstly, the Drip dissemination protocol fastly achieve the consistency of sensor network. It means that all sensor nodes receive the control message almost at the same time. Hence, they transmit data to sink node simultaneously. It causes the serious interference in network and increases the packet loss rate of network. Other than that, LEAF transmits the control message to sensor nodes continuously, message-by-message, sensor node-by-sensor node. Therefore, sensor nodes do not transmit data at the same time. As a result, there are no serious interference in network and the packet delivery ratio will be increased.

Second, the Data Type Control module of LEAF optimizes the length of data packet. It is obvious that longer packets have higher loss rate. We can see in test case 3 and 4, the packet delivery ratio of CTP+Drip combination lower than the result in test case 1 and 2. In the third and the fourth test case, the packet size in test case of CTP+Drip increase 4 bytes because all of sensor nodes have to work in the same way. LEAF does not work like that. It only send the necessary data to right applications. Then, the number of

unnecessary data packet is significantly decrease.
The third reason is the effective of Sample Rate Control module of LEAF. Sensor nodes only transmit data at required time then the packet loss rate will be reduced. According to table 6.11, we can see that the packet delivery ratio of both of LEAF and CTP+Drip decrease when applications request data at different frequency. However, LEAF can receive almost of data packet while CTP+Drip loose approximately a half. Notice that although in test case 2, 4, and 5, there are no applications required 5 second packet interval, all of sensor nodes in CTP+Drip case have to work with 5 second packet interval to provide data to all of applications.

Number of Packet Retransmission

Similar to packet delivery ratio, in the perspective of number of packet retransmission, we confirm large differences in the number of retransmissions between LEAF and Ctp+Drip. The table 6.12 show the result of the evaluation of number of packet retransmission while Figure 6.7, 6.8, 6.9 depicts the cumulative distribution function (CDF) of the number of retransmission in three test cases. In these figures, the x-axis indicates the number of retransmissions and y-axis represents the CDF. Approximately 89% of packets are delivered to the sink node without involing any retransmission in LEAF while this rate of CTP+Drip is only 30% in first two test cases and 70% in the third one. The average number of packet retransmission of LEAF is about 0.13 while the result of CTP+Drip is about 2.96. This difference is really significant because retransmission signifies the overhead of data collection in the network.
We believe that CTP+Drip require the large number of retransmissions due to it's collisions problem when all of sensor nodes transmit data almost at the same time. In addition, CTP+Drip send a large number of unnecessary data packet that causes the serious congestion in network.

Table 6.12: Retransmission

Case	LEAF	Ctp+Drip
1	0.13	4.14
2	0.23	3.26
3	0.05	1.48

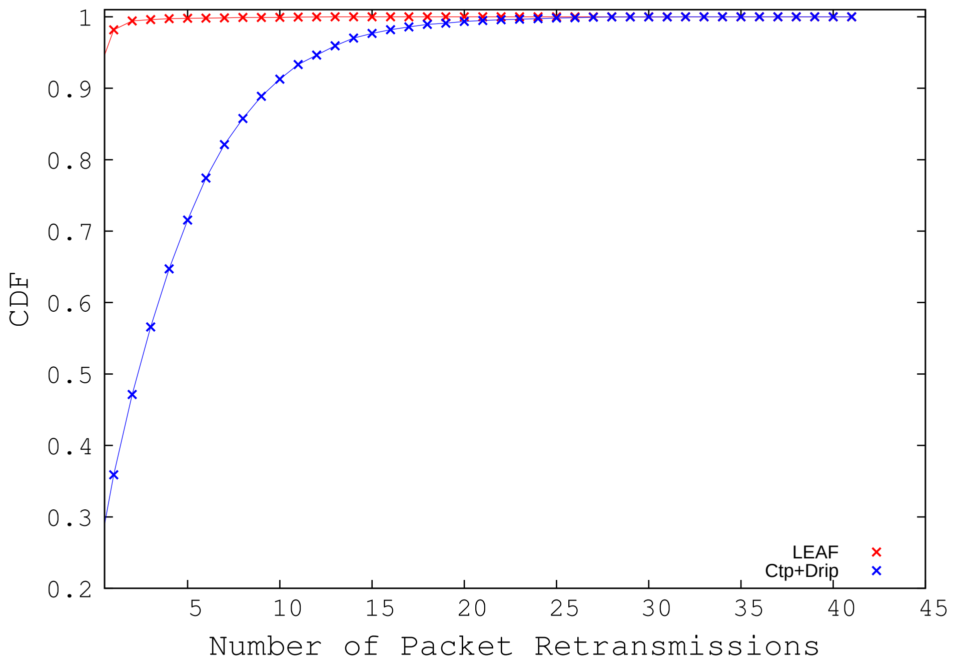


Figure 6.7: CDF of the Number of Retransmissions in the first test case

Packet Delay Variation

The figure 6.10,6.11,6.12,6.13,6.14 show the result of the evaluation in perspective of the packet delay varitaion of LEAF. In each figure, x-axis indicates the node ID and y-axis indicates the deviation between desire packet interval and the actual packet interval. Note that in the fifth test case, applications do not required data from all of sensor nodes, then in figure 6.14 only required sensor nodes information are shown. The table 6.13, 6.14, 6.15, 6.16, 6.17 show the average deviation between required packet interval and actual received packet interval. The negative values mean that data packet come to applications earlier than required while positive value have

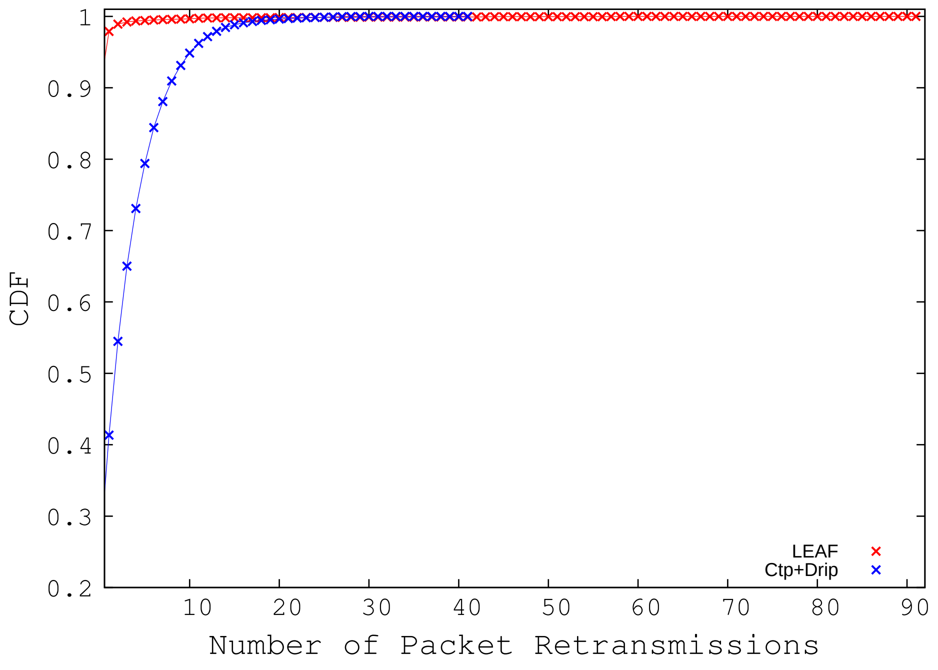


Figure 6.8: CDF of the Number of Retransmissions in the second test case

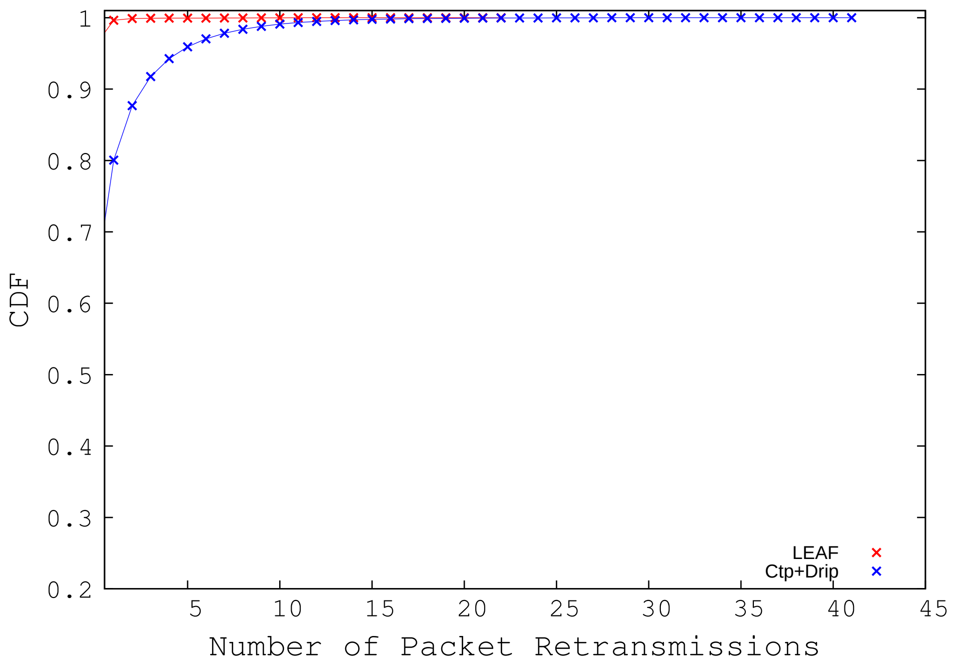


Figure 6.9: CDF of the Number of Retransmissions in the third test case

the opposite meaning. We can see that longer required packet interval is, longer deviation occurs. But in overall, LEAF can send data to applications with the variation of 3% of required packet interval.

Table 6.13: Average Packet Delay Variation in the first test case

	Required packet interval	Average Deviation
Application 1	10s	0.277s
Application 2	10s	0.305s
Application 3	10s	0.303s

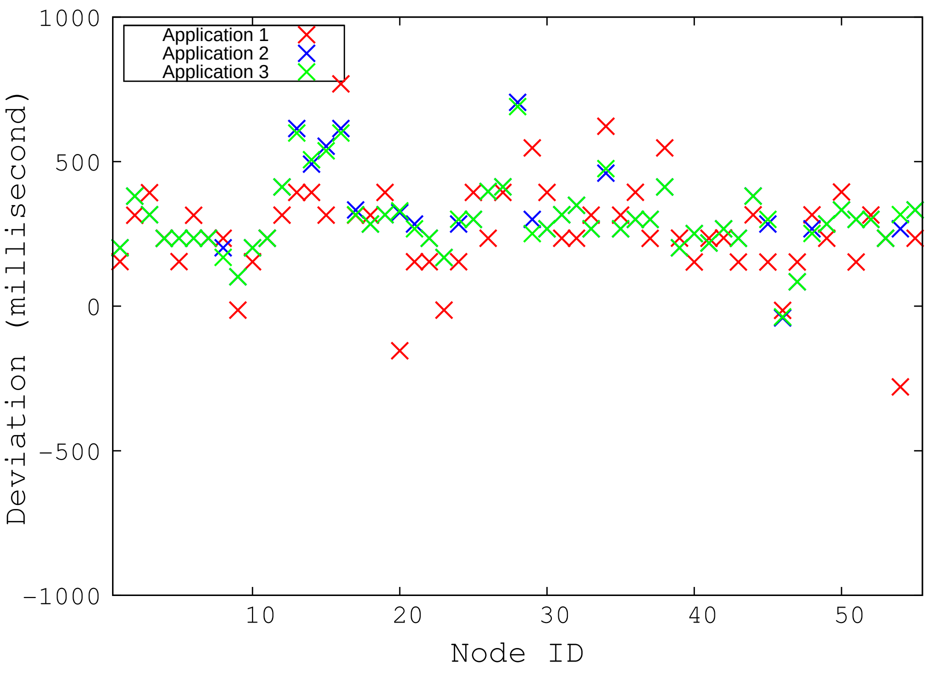


Figure 6.10: Packet Delay Variation in the first test case

Table 6.14: Average Packet Delay Variation in the second test case

	Required packet interval	Average Deviation
Application 1	15s	-0.02s
Application 2	35s	-0.305s
Application 3	60s	2.163s

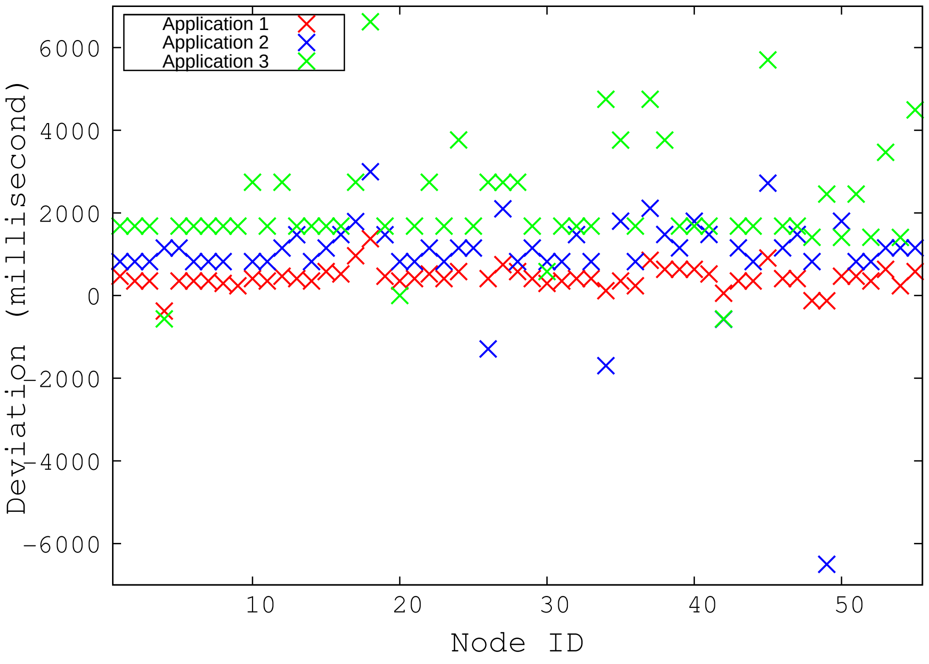


Figure 6.11: Packet Delay Variation in the second test case

Table 6.15: Average Packet Delay Variation in the third test case

	Required packet interval	Average Deviation
Application 1	20s	0.740s
Application 2	40s	1.502s
Application 3	60s	2.239s

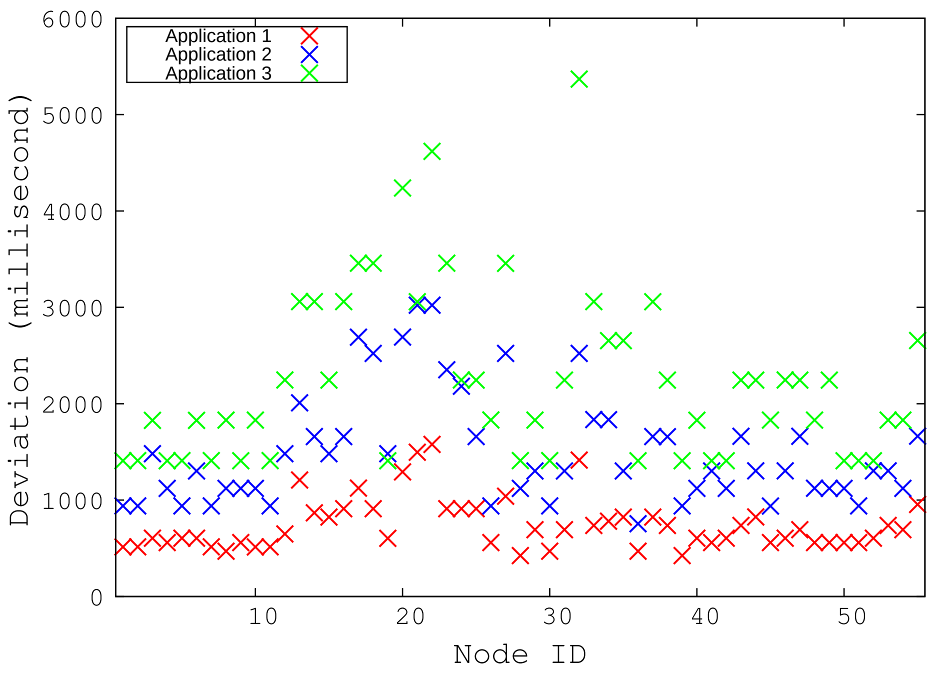


Figure 6.12: Packet Delay Variation in the third test case

Table 6.16: Average Packet Delay Variation in the fourth test case

	Required packet interval	Average Deviation
Application 1	15s	0.410s
Application 2	35s	1.500s
Application 3	60s	3.482s

Table 6.17: Average Packet Delay Variation in the fifth test case

	Required packet interval	Average Deviation
Application 1	25s	0.713s
Application 2	25s	0.720s
Application 3	15s	0.375s
Application 4	40s	1.811s
Application 5	60s	1.171s

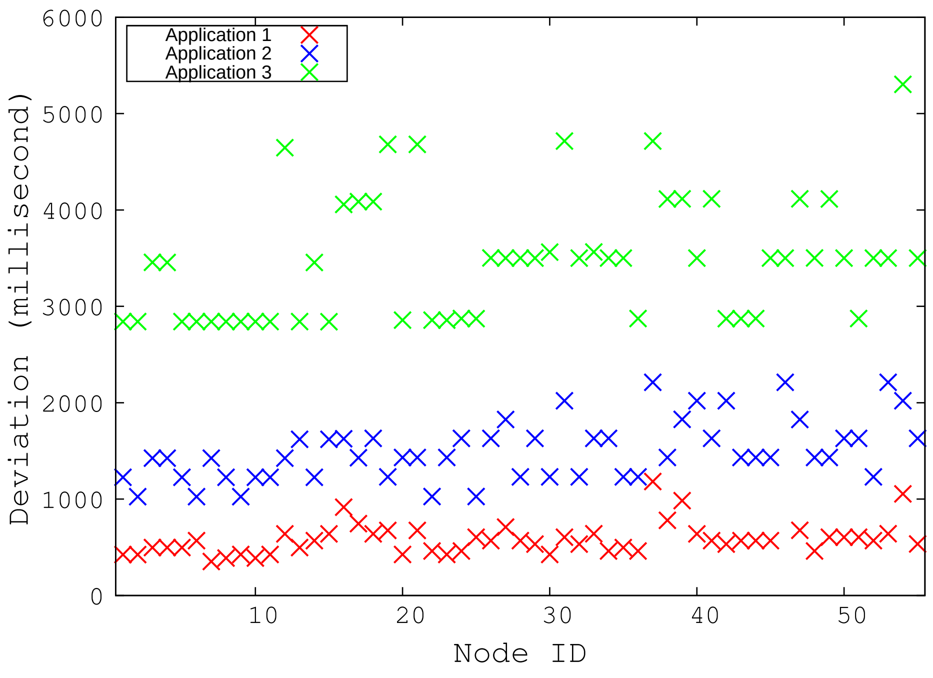


Figure 6.13: Packet Delay Variation in the fourth test case

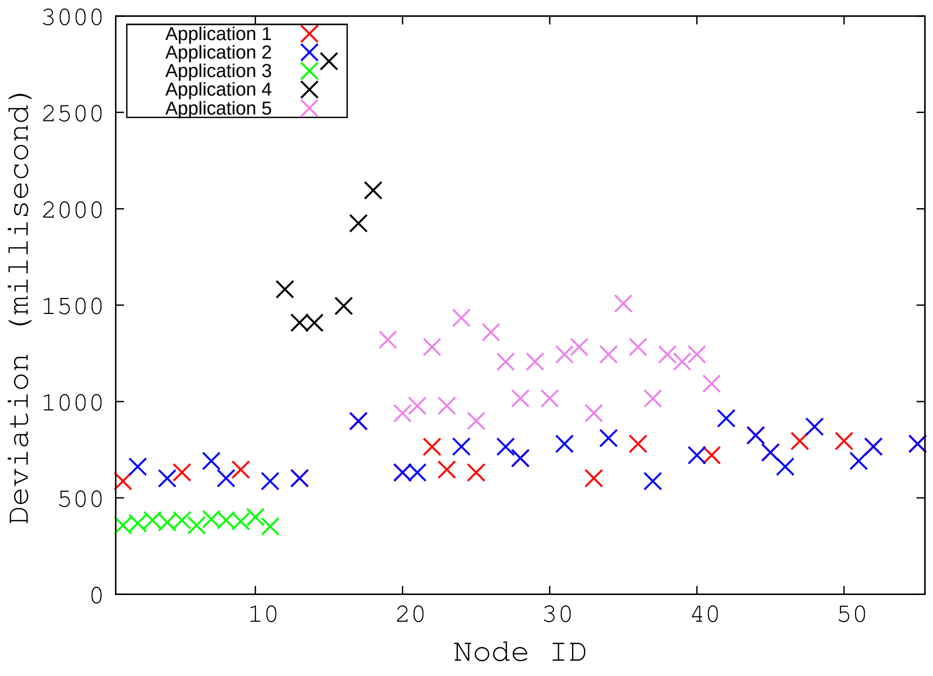


Figure 6.14: Packet Delay Variation in the fifth test case

Energy Consumption

The figure 6.15 and figure 6.16 illustrates the energy consumption of sensor nodes during 11 hours experiment of LEAF and CTP+Drip, respectively. The x-axis represents time in 1000 seconds scale, and y-axis indicates voltage in mV. Each line in these figure represents energy consumption of each sensor node. The table 6.18 show the information about time, average voltage decrement and average energy consumption of all of sensor nodes. We can see that although LEAF has more control module that make sensor nodes have to do more computation, it still has lower energy consumption than CTP+Drip. The reason is the difference of number of transmission between two method. Since sensor nodes almost use its energy on radio transmission, the reduction of number of transmission of LEAF make sense. But, in overall LEAF still does not have the high energy efficiency as expected.

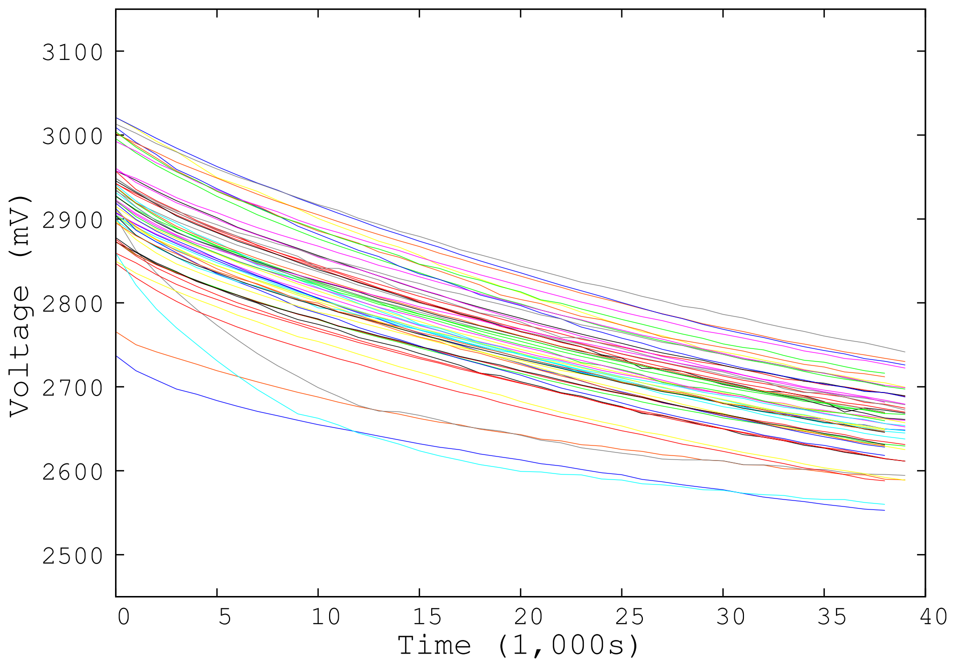


Figure 6.15: Energy consumption of LEAF

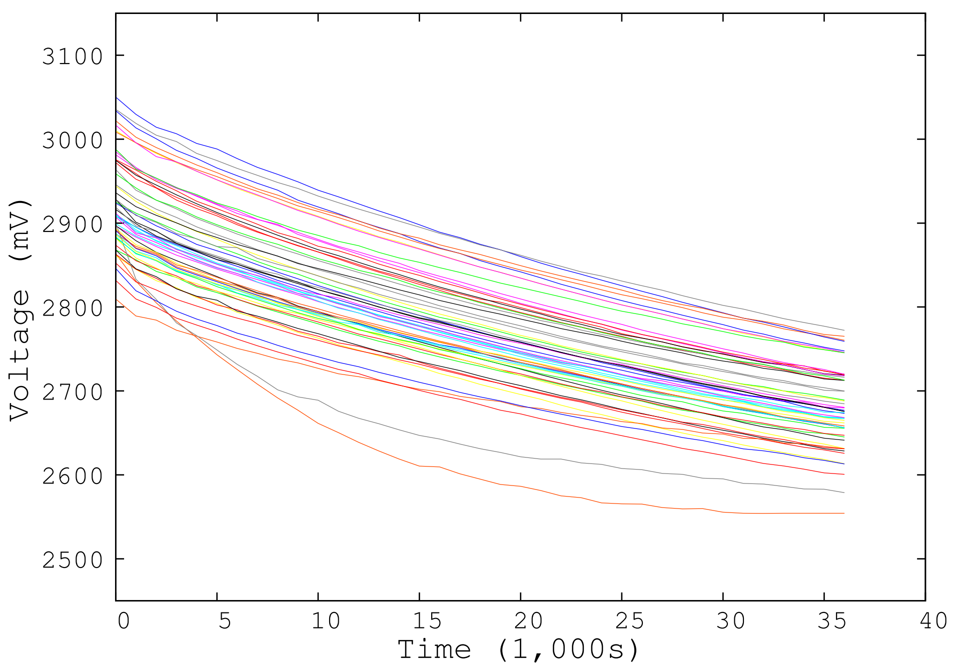


Figure 6.16: Energy consumption of CTP+Drip

Table 6.18: Energy consumption

	Average Voltage Decrement	Average Energy Consumption
LEAF	314775μV	7.823μV/s
Ctp+Drip	287674μV	7.855μV/s

Storage Usage

The table 6.19 shows the storage usage of LEAF and CTP+Drip in evaluation setup. Notice that the Control Message Queue and Serial Message Queue only available in sink node. Since LEAF has more number of component than CTP+Drip, it is obivious that LEAF use more storage than CTP+Drip. However, by using LEAF, the typical application developers do not need to add more program code into sensor nodes. Moreover, LEAF still successfully adapt to the limit memory of sensor nodes. By these reason, LEAF completely can be used.

Table 6.19: Storage Usage of LEAF and Ctp+Drip

	LEAF	Ctp+Drip
Sensor node	ROM: 39298 bytes RAM: 5512 bytes	ROM: 37802 bytes RAM: 2489bytes
Sink node	ROM: 39338 bytes RAM: 6889 bytes	ROM: 37802 bytes RAM: 6881bytes
Control Message Queue Size	20	0
Control Message Forwading Queue Size	7	0
Serial Message Queue Size	45	80
Data Message Forwarding Queue Size	35	35

6.3 Summary

In this chapter, the evaluation of LEAF is presented. We have evaluated LEAF in many test cases to confirm the ability of LEAF to become a solution of management problems in MA-WSNs. Through the evaluation, we confirmed the high packet delivery ratio (approximately 99%) with low number of packet retransmission (about 0.13) of LEAF. We also shown that LEAF can serve applications's request with only 3% deviation when compare to application requirements of packet interval. LEAF also use less energy than the combination of CTP and Drip protocols. By these reason, we can conclude that LEAF outperfoms CTP+Drip to become a solution of the management problem in Multi-Apllication Wireless Sensor Networks.

Chapter 7

Future Work and Conclusions

This chapter presents the future work and conclusions of this thesis.

7.1 Future work

At this point, LEAF uses CTP as its data collection protocol. However, the CTP data collection protocol cannot work with high sample rate. By decreasing the amount of calculation in CTP, we believe that we can achieve high sample rate data collection protocol to complete the management method of Multi-Application Wireless Sensor Network.

LEAF still have problems of storage usage and the bottle neck problem in forwarding receive message to personal computer through serial communication port. Therefore, as the future works, we have to decrease the memory used in LEAF as much as possible. We will also regulate the number of simultaneous packets inside the network to avoid the bottle neck problems by using some of efficient aggregation techniques.

At the present, LEAF is still a local wireless sensor network management method. As a future work, by connecting LEAF with some of global wireless sensor network infrastuctures, we will extend LEAF to become a global sensing infrastructure.

7.2 Conclusions

In this thesis, we proposed LEAF, a management method of Multi-Application Wireless Sensor Networks. It consits of flexiblitity protocols that can serve multiple applications request and effectively manage the resource of network to support multiple concurrent applications in a single sensor network.

By using LEAF, a sensor node's behavior is optimized. It only sense and transmit data when nescessary. Moreover, sensor nodes do not sense all of types of data, instead, based on the requests from applications, sensor nodes only measure the required kinds of data. We also built an efficient routing scheme that can choose path from sink node to sensor node and vice versa. LEAF uses a modified version of the Collection Tree Protocol as its data collection protocol.

We implemented LEAF on TinyOS and evaluated in real world environment. We built a sensor network consits of 55 Iris motes to evaluate the ability of LEAF on adapting to the various type of applications and their request. We achieved the packet delivery ratio of approximately 99%, which is 37% better than the results of the combination of CTP and Drip, two widely used protocol in current WSNs. Besides, LEAF can transmit data with the number of packet retransmission about 0.13 compare to 2.95 of related works. Moreover, LEAF can serve application request with the packet delay variation about 3% of desired packet interval.

Through careful evaluation, we believe that LEAF completely can be a solution of management problems in Multi-Application Wireless Sensor Network.

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisor, Professor Hideyuki Tokuda. This thesis would not be as it is without his technical and professional advice, guidance and unparalleled efforts in providing a magnificent working environment for me as well as all students in Tokuda Laboratory.

I would like to acknowledge and thank to Professor Jun Murai, Associate Professor Hiroyuki Kusumoto, Professor Osamu Nakamura, Associate Professor Kazunori Takashio, Assistant Professor Noriyuki Shigechika, Associate Professor Rodney D. Van Meter III, Associate Professor Keisuke Uehara, Associate Professor Jin Mitsugi, Assistant Professor Jin Nakazawa and Professor Keiji Takeda for creating a great research environment and their valuable comment.

I am extremely thankful for Mr Kenji Yonekawa, my advisor, for his great support and advice. I could learn from him network knowledges, unix skills, and how a proffesional works.

I would also want to thank Mr Naoya Namatame and Mr Ito Tomotaka for their great support and care since I joined Tokuda Lab. It is wonderful for undergraduate student like me to work with big brothers like them.

I would like to thank Mr Takuya Takimoto for his great help on my Japanese problems.

Much thanks go to members of LINK and KMSF research group in Tokuda Laboratory as well as all of members of Tokuda Laboratory for their great friendship and support. Doing research with them is one of my most memorable time in my life. Thank to Kohei Saijo for sharing sleepless nights in lab with me during this thesis.

I extremely thankful for Mrs Yuka Tatebayashi Ms Sachi Suzuki for their great suppor and encouragement. They treated me as if I were their family. Through them, i could learn many things about Japanese culture.

I would also like to thank my friends from Vietnamese-Keio group for their great support and sharing in daily life in Japan.

Lastly and most importantly, to my family. I appreciate my beloved parents, little sister for their never-ending and unconditional love, support and encouragement.

LEAF, name of my thesis, is English meaning of my beloved little sister's name.

Nguyen Gia
February 14, 2012

Bibliography

[1] Mark Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7):75–84, 1993.

[2] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 88–97, New York, NY, USA, 2002. ACM.

[3] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 13–24, New York, NY, USA, 2004. ACM.

[4] Tian He, Sudha Krishnamurthy, John A. Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Jonathan Hui, and Bruce Krogh. Energy-efficient surveillance system using wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pages 270–283, New York, NY, USA, 2004. ACM.

[5] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605 – 634, 2004.

[6] Konrad Lorincz, Benjamin Kuris, Steven M. Ayer, Shyamal Patel, Paolo Bonato, and Matt Welsh. Wearable wireless sensor network to

assess clinical status in patients with neurological disorders. In *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, pages 563–564, New York, NY, USA, 2007. ACM.

[7] R. Jafari, A. Encarnacao, A. Zahoory, F. Dabiri, H. Noshadi, and M. Sarrafzadeh. Wireless sensor networks for health monitoring. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pages 479 – 481, july 2005.

[8] Judy York and Parag C Pendharkar. Human—computer interaction issues for mobile computing in a variable work context. *International Journal of Human-Computer Studies*, 60(5-6):771 – 797, 2004.

[9] Randall B. Smith. Spotworld and the sun spot. In *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, pages 565–566, New York, NY, USA, 2007. ACM.

[10] Michael Beigl, Christian Decker, Albert Krohn, Till Riedel, and Tobias Zimmer. parts: Low cost sensor networks at scale. In *Proceedings of The Sevent International Conference on Ubiquitous Computing*, UBICOMP '05, Tokyo, Japan, 2005.

[11] Yang Yu, Loren J. Rittle, Vartika Bhandari, and Jason B. LeBrun. Supporting concurrent applications in wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 139–152, New York, NY, USA, 2006. ACM.

[12] P.J. del Cid, S. Michiels, W. Joosen, and D. Hughes. Middleware for resource sharing in multi-purpose wireless sensor networks. In *Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on*, pages 1 –8, nov. 2010.

[13] Amjed Majeed and Tanveer A. Zia. Running multi-sequence applications in wireless sensor networks. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '09, pages 365–369, New York, NY, USA, 2009. ACM.

[14] Huang Lee, Abtin Keshavarzian, and Hamid Aghajan. Near-lifetime-optimal data collection in wireless sensor networks via spatio-temporal load balancing. *ACM Trans. Sen. Netw.*, 6:26:1–26:32, June 2010.

[15] Jian Li, A. Deshpande, and S. Khuller. On computing compression trees for data collection in wireless sensor networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[16] Tao Shu, M. Krunz, and Sisi Liu. Secure data collection in wireless sensor networks using randomized dispersive routes. *Mobile Computing, IEEE Transactions on*, 9(7):941 –954, july 2010.

[17] Shuai Gao, Hongke Zhang, and S.K. Das. Efficient data collection in wireless sensor networks with path-constrained mobile sinks. *Mobile Computing, IEEE Transactions on*, 10(4):592 –608, april 2011.

[18] D. Chu, A. Deshpande, J.M. Hellerstein, and Wei Hong. Approximate data collection in sensor networks using probabilistic models. In *Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on*, page 48, april 2006.

[19] Chong Liu, Kui Wu, and Jian Pei. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *Parallel and Distributed Systems, IEEE Transactions on*, 18(7):1010 –1023, july 2007.

[20] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM.

[21] R Fonseca, O Gnawali, K Jamieson, and P Levis. Four-bit wireless link estimation. In *Proceedings of the 6th Workshop on Hot Topics in Networks*, HotNets-VI, New York, NY, USA, 2007. ACM SIGCOMM.

[22] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, pages 2–2, Berkeley, CA, USA, 2004. USENIX Association.

[23] Naoya Namatame, Jin Nakazawa, Kazunori Takashio, and Hideyuki Tokuda. Sensingcloud: Open and global sensor network using distributed aggregation mechanism. *Ubicomp in the Large: Collaborative Sensing and Collective Phenomena*, pages ppNA–ppNA, 5 2010.