Keio University Master's Thesis Academic Year 2010

# User Driven Code Propagation Mechanism for Urban Sensor Networks

**Keio University Graduate School of Media and Governance**

**Takatosi Kanazawa**

修士論文 2010 年度 (平成 22 年度)

# アーバンセンサネットワークにおけるユーザ指向のアプリケーション動的更新機構の構築


## 論文要旨

既存のセンサネットワークアプリケーションの多くは特定の環境下であらかじめ規定された動作を行うことが想定されている．しかし，マイクロプロセッサ技術の発展によるセンサデバイスやネットワークのエンドユーザが携帯する携帯端末の高機能化により，近年は市街地等における，より広範囲における汎用的なセンシングを行うことに関する研究が盛んである．このような環境を対象としたセンサネットワークはアーバンセンサネットワークと称される．

アーバンセンサネットワーク上でセンシング基盤がユーザのセンシング要求を満たすことにおける重要な課題として，複数のセンサノードの中からセンシングに用いるノードの決定，及び選択されたセンサノードにセンシングを行うアプリケーションの動的挿入があげられる．これに対して，既存研究においてはセンシングの機会を動的に検索する機会センシングの概念や，階層化されたセンサノードアーキテクチャを用いたネットワーク構成に関する研究が多く提案されている．一方で，既存の多くのアーキテクチャはネットワークの中心を電力や有線ネットワークが整備された地域に持つ中央集権型のアーキテクチャで構成されており，ネットワークの端に位置するユーザの要求に対するセンシング機会の検索及びアプリケーションの挿入は低速である．

本研究ではアーバンセンサネットワークのエッジネットワークにおけるユーザ指向の動的アプリケーションコード挿入を効率的に行う為の機構を提案する．環境としては電力や有線ネットワーク等既存のインフラが整備されておらず，ネットワークへのユーザアクセスが限られるようなセンサネットワークにおいてユーザが自身の近隣をセンシングする要求がある場合を想定する．このような環境においてユーザは携帯デバイスを通して近隣のセンサノードを検索し，センサノードに対して自身の要求を満たすことが可能であるアプリケーションを直接挿入することによる効率的なセンシングを実現する．本研究は実装基盤として IRIS を用い，これを実環境に 18 個配置し，アーキテクチャにおけるノードのローカライゼーションとアプリケーションコードの挿入の正確性と遅延に関する定量的評価を行った．評価結果を通して，アーバンセンサネットワークのエッジネットワークにおける本機構を用いた近接ノードのローカライゼーション及びアプリケーションコードの挿入を十分な精度及び速度をもって行うことが可能であることを示した．



**キーワード:**

1 アーバンセンサネットワーク　2 アプリケーションの動的更新

3 ローカライゼーション　4 ネットワークアーキテクチャ



**慶應義塾大学大学院政策・メディア研究科**

**金澤 貴俊**

**Abstract of Master's Thesis Academic Year 2010**

# User Driven Code Propagation Mechanism for Urban Sensor Networks


## Summary

Traditional sensor network applications have focused on deployment for a specific purpose to perform predefined tasks. On the other hand, advances in microprocessor technology has greatly improvised the sensing capability of sensor devices both on the user and infrastructural side, and has shed light to sensor network deployment in a broader, more complicated area such as in a city environment. Such sensor networks are often referred to as an urban sensor network, where the sensor network's capability not only covers the traditional, application specific domains of sensing, but also able to directly serve the user requirements of sensing through the infrastructure.

The core requirements for an urban sensing architecture is for the architecture to dynamically and rapidly serve the spontaneous user's sensing request through sensor resource allocation and application propagation. Several architectures have been proposed in order to perform urban sensing, where the core architecture evolves among the principle of opportunistic sensing, which is a principle to search for dynamic changing sensing opportunities over time, and the network components structured in a hierarchical manner. However, most existing urban sensing architectures assumes a centralized approach when probing for sensing opportunities. Therefore among the edge networks within the urban sensing environment where the network is located distant from the network's core, the above principle and architecture could become a bottleneck for user driven application code propagation into the sensor network.

In our work, we propose an efficient user driven code propagation mechanism for edge network within the urban sensor network. Our work assumes an user who wishes to sense his environmental information on a edge network within the urban sensor network, where the existing infrastructural facilities are insufficient and user access to the area is limited compared to the central areas of the urban environment. On sensing requirements from the user, our mechanism first localizes the nearest nodes around the user, and directly propagates application onto the localized node.

We have implemented and evaluated our work under a real life environment using 18 sensor nodes. Our experimental results suggests the applicability of our user driven code propagation mechanism under certain scenarios, and have proved the applicability of our direct user driven code propagation scheme for future urban sensing architectures where the environmental characteristics vary over sensing areas.



## Keywords:

1 Urban Sensor Networking  2 Application Reprogramming

3 Localization  4 Network Architecture



**Keio University Graduate School of Media and Governance**

**Takatosi Kanazawa**

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Traditional sensor network researches have focused on network deployment for special purpose applications to perform sensing task where human reach is difficult or impossible.

On the other hand, the emerging propagation of rich communication infrastructure in urban areas and the people living in such areas being able to exploit such infrastructures through smart handheld gadgets, has shed light to sensor network deployment in urban areas, often referred to as an urban sensor network. A key focus of urban sensor network is for the sensing infrastructure to not only perform preemptive sensing tasks, such as acquiring of the temperature within the sensor network's coverage area, but rather the infrastructure to collaborate with the people living in such environments. This collaboration occurs in two-fold; the sensor network infrastructure could exploit the people's communication devices, in order to achieve functionality statically deployed sensor networks are not capable of, or people could query the sensor network to acquire information of interest. Therefore, in our work, we specifically refer to such sensor networks as people centric sensor networks (PCSNs).

Combining statically deployed sensing infrastructure and dynamic moving humans give several merits compared to statically deployed sensor networks. Assume two sensor networks deployed in distant areas where data

transmission is impossible between the networks. If the mobile nodes were to traverse between the two regions and were able to communicate with the two networks, the mobile node could mule data between the two networks to interconnect the networks otherwise impossible without the mobile node. Dynamic sensing area coverage is also an advantageous factor of PCSNs. While statically deployed sensor network coverage is limited to its initial deployment, if the mobile nodes were to have connectivity with the network, the sensing coverage area is basically unlimited based on the mobile nodes movement perimeter. On the other hand, PCSNs require the flexibility to satisfy the user's various sensing requirements, and statically deployed sensors are not uniform, but rather heterogeneous to meet this requirement. Therefore, the infrastructural architecture is required to masquerade the underlying heterogeneity through multiple layers. For example, if the user wishes to sense a certain area for a certain time period, the infrastructure would be required to materialize the user requirement such as by checking the types of sensors required for the sensing task, the available sensors within the time frame, and the reliability of the data.

In our work, we extend the notion of PCSNs, especially described by the MetroSense project[7]. MetroSense applies a tired architecture, where the components of the sensor network is classified by three components, the server tier, the sensor access point tier, and the sensor tier. The sensor tier collects the sensing data required for various tasks which is collected by the sensor access point tier which maintain the functionality of a sensor network gateway. The components of the sensor access point tier is interconnected to the server tier, which offers the core functionality of the infrastructural management of the PCSN. Further details of MetroSense is described in Section 3. In our research, we assume the heterogeneity of the sensing infrastructure fidelity, i.e., the infrastructural deployment density of sensor and existing infrastructure such as networks and power lines are not uniform, such as in a metropolitan city area and a natural park located in the edge of the PCSN. Under this scenario, we present three core problems which arise if applied

current PCSN architectures. First of all, the application code propagation overhead is closely related to the heterogeneity of the infrastructure. Under this infrastructure, the opportunistic code propagation model applied by existing research is verbose because the user's sensing requirement could be fulfilled only by sensor and human interaction. Data acquisition is also slow because the data needs to be relayed to intermediate gathering nodes.

We challenge the problems mentioned above by proposing a user driven code propagation model for acquiring sensor data. In our model, users would priory download application code required for sensing through the rich infrastructure, and on occasions where users would wish to sense data where the underlying infrastructure is poor, users would dynamically propagate application code to his nearest sensor nodes which enables acquisition of sensor data. Our model cuts the overhead of user query traversal through existing urban sensing infrastructure and users directly interacting with nearby sensors on querying request (application data propagation and querying requirement to the application), and sensor data acquisition from the sensors.

We have implemented our model on the TinyOS[23] software platform using the IRIS hardware, and evaluated our work under a real world environment inside the Keio Shonan Fujisawa Campus using 18 sensor nodes. Through our evaluation, we have proved the applicability of our architecture under our assumed environment.

The rest of our work is organized as follows. Chapter 2 discusses the background details of urban sensing through existing researches and key characteristics. In Chapter 3, we will describe the use-case scenario and environmental assumption of our proposed code propagation mechanism, and Chapter 4 describes the core architecture of our proposal. Evaluation of our work on a real life environment is shown in Chapter 5, and our concluding remarks and future works are discussed in Chapter 6.

# Chapter 2

# People Centric Sensor Networks

In this chapter, we describe the principles, architecture and enabling technologies of PCSNs through existing researches, and define our target environment and problem statement.

## 2.1 Sensor Network Research

Many of the existing researches of sensor network technology has focused on deployment in a specific environment to execute predefined tasks. In many cases, acquisition of data among the deployed environment is the first priority task of the sensor network. Such deployment of a sensor network is often referred to as a data-centric architecture, where the architectural improvements made were mostly for the purpose of improving performance of data acquisition. Below we summarize the specific purpose sensor network applications among various deployment fields.

- **Agricultural :** Deployment of a sensor network in an agricultural environment provides several merits, due to the vast area required for agriculture, and necessity of event detection among an identical landscape. The Lofar Agro project[2] has deployed a sensor network in a

crop field for precision agriculture, which focuses on monitoring microclimates in a crop field. Sensor network deployment in a vineyard is discussed in [6], which monitors both the state of the vineyard and the manufacturer in order to study the optimal deployment scenario given a specific environment. Kwong[21] in his work has proposed a wireless sensor/actuator network application for use in the cattle breeding industry, by deployment of sensor nodes onto a cow.

- **Military :** Because the initial deployment motivation for a sensor network has derived from military objectives, several applications exists in this field of research. Bokareva[5] proposes a ground surveillance sensor network to detect vehicles and troops in a military application, using acoustic and magnetic sensors. Simon[34] has presented a counter sniper system to locate shooters using a sensor network even in urban environments, and Arora[1]proposes an intrusion detection system by location of a breach in a certain perimeter by quantitate deployment of cheap sensors.

- **Environmental Monitoring :** Data acquisition in areas where human interference is difficult or impossible is a major characteristic of a sensor network. A great amount of deployment exists in this category; just to name a few, the PermaSense[36] project attempts to monitor the state of permafrost in the Swiss Alps, for the purpose of environmental study and avalanche prevention. On the contrary, Werner-Allen[38] has propose deployment of a sensor network in an active volcano for geographical studies using several MICA sensor nodes. Multiple deployment of sensor network in an underwater environment is proposed in [8].

In many cases, multiple sensor nodes with low capability such as limited computation power, storage capacity, and network transmission capability, are deployed among a field of interest to sample data. Communication between nodes are often unreliable due to the nature of wireless transmission,

yet applying existing transmission control schemes as applied in existing infrastructures such as TCP[30] is unacceptable due to the limited capability. Under this environment, several architectural improvement have been made. Several researches exists in the field of data routing[40][41][18][39][16][19], in order to assure data transmission with better capability than the primitive flooding technique where the node broadcasts data to its adjacent nodes.

Diverse software composition primitives also exists for the sensor network to perform specific tasks. A common approach is to modularize the necessary components before installation onto the sensor nodes to reduce the total software size. TinyOS[24] is a commonly used software primitive which is written in nesC[12] and applies a component based architecture with an event driven execution paradigm. Contiki[10] enables dynamic loading of software modules in a standard ELF format over the network, and introduces proto-threads , an abstraction of a thread-like programming model with minimal memory overhead. Mantis[4] and Nano-RK[11] is based on preemptive multithreading, which unlike event driven primitives allots a time frame for each thread and the kernel decides the execution thread.

## 2.2 People Centric Sensing

However, drastic advances in technology both on the infrastructural and the end user's handheld gadgets have risen novel opportunities to bring sensor network technology centered around the presence of humans. Human centric sensing is a notion adopted by the MetroSense architecture[7], where the sensor network would sense data for people queried by people themselves, or of people where the environment would sense humans for multiple opportunities. PCSNs differ from traditional sensor networks in characteristics with the introduction of humans, not only as a sensing target, but also as an infrastructural component to preform important tasks within the network.

## 2.3 Existing PCSN Architectures

Several architecture have been proposed in effort to bring existing sensor networks to urban environments. Our work has heavily been affected by the MetroSense project[7], an attempt to design a sensing platform in urban environments where the architectural design grows around the interaction of static and mobile sensors. The core design principles of the MetroSense architecture include network symbiosis with existing infrastructures, resource asymmetric aware design where the composing sensor nodes are classified into specific tiers based on its role within the network, and localized interaction for avoiding complex multi-hop interactions. The tiered architecture is composed of a server tier which manages the central architectural controlling of the network, the sensor access point (SAP) tier which serves as an intermediate access point and gateway for mobile and static sensor nodes, and the sensor tier composed of mobile and static sensors. The MetroSense architecture exhibits the principle of opportunistic sensor networking, which extends the sensing capability of a network via collaboration of mobile and static nodes. For example, a query to sense an environment without static sensor deployment could be achieved by a mobile node with its traversal path overlapping in the area of interest, in the specific time window the user wishes to acquire data. The SenseWeb[17] architecture describes a deployment scenario with similar architectural characteristics as the MetroSense architecture. Three core components, namely a sensor gateway which acts as a gateway for statically deployed sensors, mobile proxy with the same functionality as the prior but applies to mobile nodes, and the coordinator which handles the central administration within the network, are interconnected via API in order to achieve sensing in an urban atmosphere. The SenseWeb infrastructure could be access through an web interface in order to perform various sensing tasks, such as visualization of historic pollution distribution within a city from car-mounted devices, a debris flow monitoring and warning application and national scale weather monitoring. Finally, the CitySense[26] project proposes an attempt to construct an efficient testbed

infrastructure for sensor network deployment in an urban atmosphere.

# Chapter 3

# Use-case Scenario and Environment Assumption

In this chapter, we first give an example use-case scenario of our work, and summarize our environmental Assumptions.

## 3.1 Use-case Scenario

Assume an outdoor environment located at the edge of the urban atmosphere; supposedly a national park with a decent floor space. Under this environment a sensor network is deployed, in which it's mission is to support and inform users of their surrounding events (Figure 3.1). For convention, we refer to this area as National Park SFC (NPS).

Now assume an elderly woman, Alice, whom finished his afternoon teatime with his fellow elders, and decided to take a walk in the NPS. Her recent concerns are; too much exposition to the sun which might result in ultraviolet ray poisoning, bird watching, and life logging, an attempt to log his environmental data which is recommended by his physician for planning his future treatment. Our attempt here is to solve his concerns by his interaction with the PCSN deployed in the NPS. Each of her matters could be dealt with the following PCSN applications described in table 3.1.

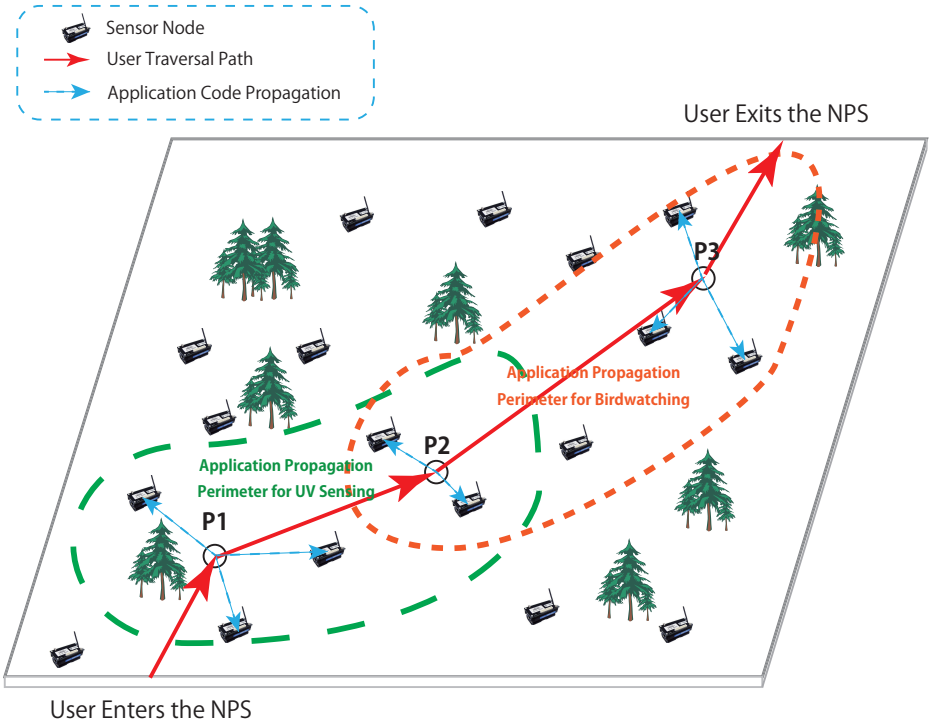


Figure 3.1: Use-case Scenario

Table 3.1: Correspondence of User Requirements and Enabling Applications

| User Requirement | Enabling Application |
|---|---|
| Ultraviolet Ray Avoidance | Violet Ray Sensing and Guidance |
| Bird Watching | Movement Sensing |
| Life Logging | Environmental Sensing |

Alice has entered the NPS at P1 the time is 2PM, and her main concern is to avoid the ultraviolet rays within her traversal path. She would then issue a query through her smartphone, to sense the ultraviolet ray near her current position, to avoid direct sunlight and walk through the shady path. Internally, the smartphone would localize the nearest sensor nodes, and propagate an application code to the localized nodes. After code propagation is done, her nearest nodes would sense and send the ultraviolet ray data towards her smartphone. During her time span of concern towards her ultraviolet ray poisoning, the smartphone would continue to issue query to nearest nodes even while she is moving, and update the most local nodes and propagate application code, and the application on each sensor node would time out after a specified time interval. At P2, the sun has shaded into the clouds, moderate temperature among her surroundings, and Alice has decided to enjoy birdwatching. She would then issue a query to birdwatch in the same manner as sensing ultraviolet rays, and this time the application would sense the movement of nearby creatures, in which the direction of the movement is sent to her. All this time, her life logging application has also been running, i.e., localized and application code propagated, in the background in the same manner as the previous two applications, which enables her collecting of the necessary information.

## 3.2 Problem Definition

In our scenario, we assume the PCSN to serve the user's imminent request for sensing environmental data around the user by direct localization and

code propagation to the users nearest node, in effort to efficiently selecting the necessary nodes for sensing and propagation code over a minimal hop count. The sensor network is located at the edge of an urban sensor network where human access is rare, and are not able to improve communication reliability or the lifespan of the sensor nodes through exploiting existing infrastructures. That being said, existing human sensing architectures are able to cope with this scenario through its own functionalities, but are inefficient specifically because of the heterogeneity of the deployment environment; . Below we summarize the concerns which would arise under our example scenario, specifically on the MetroSense and SenseWeb architecture.

1. **Significant Code Propagation Overhead to Distant Networks :** MetroSense and CitySense assumes a tier architecture for network composition. The crutch of this architecture among application code propagation is the intermediate access points (SAP in MetroSense, mobile proxy and sensor gateway in CitySense). If applied to our scenario, among code propagation, the user query would first be accumulated in the management domain of the architecture, where it would sense for opportunity of available sensors for application propagation. The application would then be delivered to the intermediate nodes, where it would install sensors to the opportunistic nodes. This would be non-problematic if the intermediate nodes were to be consistently deployed among an area. However the case is rare because a city consists of areas of various geographical characteristics, making deployment of sensor nodes in a certain area difficult than others. For example a downtown location would be an easy deployment environment due to the underlying infrastructure and ease of access for node maintenance, but a national park located at the edge of the town would be the opposite due to accessibility and human traffic. Assuming this heterogeneity of environmental characteristics correlate with node deployment, delivering an application image with a significant

size of data to an edge network would obviously be problematic because the quality of intermediate path for data delivery dissipates as the opportunity for sensing shifts to the edge network.

2. **Redundancy of Opportunistic Operations:** The existing architectures provide opportunistic sensing capability for sensing, which includes operations such as resource allocation to available sensors and opportunistic delegation. This functionality exists because the principle of the architecture is to utilize static and mobile sensor activity in order to achieve better sensing capability. However, among the above scenario, the necessity for opportunistic operations are limited to the availability of resources among the sensors at best because the query derives one way from the mobile user to static sensors., which makes opportunistic probing of geographically available sensors are unnecessary. As mentioned in our previous problem statement, it is desirable to avoid redundant traffics at the edge of the human centric network because of the great communication overhead.

In short, edge networks within the human centric network suffer from network communication overhead because of the geographical heterogeneity.

## 3.3 Our Architectural Proposal

In our work, we present a user driven code propagation mechanism to solve the problems mentioned in the previous section. As in the use-case scenario, our architecture takes part as an individual network within the human centric sensing environment such as mentioned in the MetroSense architecture. Within our proposed network, the user interacts with the static deployed sensors in the environment, directly, through a portable device. Application code the user wishes to propagate are pre-installed onto his portable device through a rich infrastructure, which is brought inside the edge network for installation. Our architecture addresses the above problems because the opportunistic operations reside inside the network without interaction with a

central identity.

## 3.4 Environmental Assumptions

In this section we summarize the environmental characteristics assumed in our work.

### 3.4.1 Geographical Characteristics of the PCSN

Our architecture assumes a PCSN where its deployment purpose is expected to satisfy a user's sensing requirement within the network, but is geographically located at the edge of the urban atmosphere, such as the example shown in our use-case scenario. Contrary to PCSNs where existing infrastructures such as power lines, wired communication backbones, and other specifically deployed sensor networks, are free to exploit for the network's stability and better performance, our target environment lacks such features, therefore with limited capability.

### 3.4.2 Sensor Network Deployment

As mentioned in the previous section, the PCSN we target cannot exploit existing infrastructures, but are instead required to self-manage the network architecture and resources of each sensor nodes, such as the node's power supply. We also target an outdoor environment with a large area, such as forests or natural land sights, which requires a significant amount of sensor nodes for area coverage. Therefore, our requirements for sensor network deployment are i) deployment of multiple **cheap** sensor nodes to quantitatively deploy in a large area, and ii) an **energy-aware sensing platform** without need for relying on existing infrastructures.

### 3.4.3 Application Reprogrammable Infrastructure

Our architecture focuses on dynamic serving of user request issued towards the sensor network. Assuming multiple users each with his own requests

towards the network, the infrastructure is required to host multiple applications corresponding to user request. A naive solution would be to predict the user requests which might be issued in the future, and store the applications to match the request inside the network. However, our assumption of a sensor node lacks the storage capacity to hold several megabytes of application data. Therefore, we assume an application reprogrammable infrastructure for dynamic interaction with the users in order to leave potential for serving future applications.

# Chapter 4

# User Driven Code Propagation Mechanism

In this chapter, we describe our proposed architecture in detail. First we describe the overview of our architecture, followed by description of our core mechanism, node localization and code propagation.

## 4.1 System Overview

Figure 4.1 describes the system architecture diagram of our system. Our system consists of the **propagation node** and the **sensing node**, where the propagation node is the user's handheld device, and the sensing node resides within the environment.

The propagation node consists of mainly three components, the system management interface, the node localization module, and the application code propagation module. The system management interface receives application query from the user, which then calls the node localization module for nearest node localization. The localization module then sends an RSSI query message over the radio to the sensing nodes which sends an RSSI acknowledgment packet to the propagation node. The acknowledgment packet is then queued in a buffer of size N, and the x number of highest RSSI val-

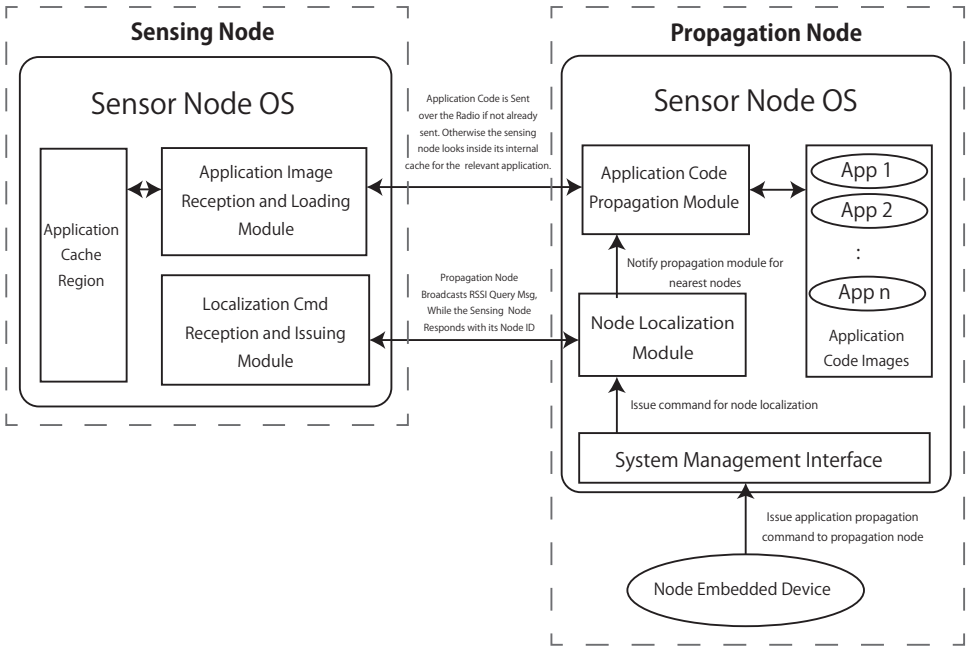


Figure 4.1: System Architecture

ues and its corresponding nodes are selected as the candidate node for code propagation. This information is then sent to the application code propagation module, which is used to issue application propagation commands to the target node. The localized node which has received the propagation command would then act as follows; a) if the application for propagation have not been previously installed on the sensing node, the node would issue a binary code image transmission request to the propagation node, or b) if the application has already been previously installed on the sensing node and the code image still resides inside his cache region, the sensing node would simply load and run the application image in his cache memory. Figure 4.2 describes our code propagation flow of our architecture.

## 4.2 Design Principle

The goal of our work is to sense information around the user environment in the target PCSN area. The Deluge[15] approach of flooding the network

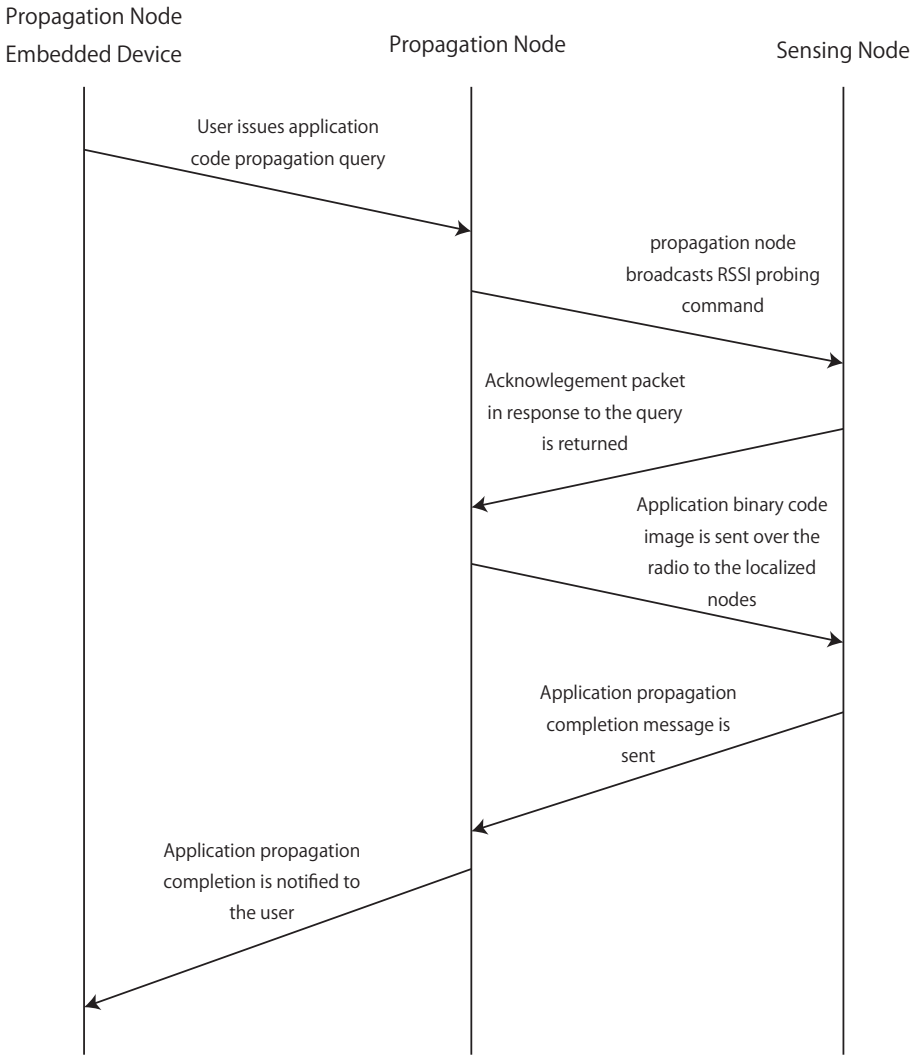


Figure 4.2: Flow Diagram

with a single application although efficient for that purpose, irrelevant in terms of our usage to sense a specified area of interest, i.e., human surroundings. The requirement for code propagation in our scenario is to multicast the necessary code image to the pre-specified node. We have achieved our goal by applying the following modification to the Deluge architecture; a) optional multicasting of propagation command to the sensor nodes, and b) self-management of application state on each sensor nodes.

**Optional Multicast**

Nodes programmed with the Deluge framework broadcasts its current state over the network for the purpose of application propagation and for confirmation of the newest application code image. In our architecture, we disable this functionality of autonomous broadcasting of control sequences over the network, and instead each node would only respond to control messages sent by the propagation node. After localization of the nearest nodes, the propagation node would queue a message corresponding to the node ID of the nearest nodes in its internal buffer, and start a timer on a 500ms interval for queue consumption. Call to the timer would cause the propagation node to unicast a code propagation control message over the radio directly to the specified node. The timer would be called until it has consumed the queue. The sensing node would return an acknowledgment message corresponding to the propagation message. If the acknowledgment packet were not to be received on the propagation node, the propagation node would retransmit the propagation packet until it has successfully acknowledged the reception of the sensing node.

**Self-Management of Application State**

Our architecture assumes the propagation node to move over time, and the nodes required for sensing at a single time frame does not change. If the sensing node would be left to be reprogrammed with the previous application until the next reception of a reprogramming query from the propagation

node, the node could continue to run its previous application even after the user has left the necessary sensing range. Therefore, in our architecture we apply a runtime counter for each sensing node, which expired would stop the execution of its current application.

## 4.3 Enabling Approaches

In this section we summarize the alternative approaches which exists for node localization and application code propagation in a PCSN, and propose our motivation for application of RSSI and the Deluge framework for system composition.

### 4.3.1 Existing Localization Methods

Localization of nearby nodes is a critical factor among our architecture because our focus is to sense the user's surrounding environment. Several approaches exists for node localization in a sensor network, as detailed in [14]. Below we summarize the localization methods used in a mobile network, and our approach for node localization.

**GPS Based Methods :** The most obvious solution localization would be for each sensor nodes to be equipped with a location tracking device such as GPS or precoded with an absolute location indicator before deployment. Niculescu[28] assumes a sensor network in which some nodes within the network are equipped with GPS receivers, while the sensor network presented by Priyantha[31] assumes some nodes in the network to be hardcoded with geographical information pre deployment. However, GPS hardware is expensive and causes significant overhead on power resources, especially considering a cheap sensor node supposed for mass deployment. The precoding of location is also ineffective in our scenario because we assume an outdoor environment where the sensor node's location could change over time, such as by natural force, wild animals, malicious users, etc.

**Range Based Methods :** The works in [33] and [34], applies localiza-

tion via time difference of arrival of two different signals (TDOA). TDOA is based on triangulation, where the node is localized via time difference of arrival of three or more nodes. Though TDOA could achieve precise localization through calculation and with the use of high-precision antennas, they often rely on expensive hardware for signal transmission, which in our case unacceptable in terms of deployment cost.

**RSSI Based Methods :** In our architecture, we localize nearest nodes by Received Signal Strength Indicator. RSSI is a metric for signal strength of a radio signal and is acquired by the signal receiving transceiver during the intermediate frequency stage before amplification. Depending on the radio transceiver used for measurement, the value would range from 0 to 255. RSSI based localization of nodes are applied in various system architectures. RADAR[3] uses the RSSI signal strength information gathered by multiple sensor nodes to triangulate the users coordinate by empirical data and theoretically computed signal strength information. The work proposed by Patwari[29] combines RSS and connectivity information.

In our architecture, we do not require a precise localization scheme, but rather sufficient with a vague localization; i.e.,acquiring the nearest nodes from the propagation node. Although RSSI values does not necessary correlate to distance between the two nodes, we show through our evaluation that RSSI is a sufficient parameter for localization in our scheme.

### 4.3.2 Existing Code Reprogramming Methods

Several architectures exists for reprogramming a sensor network, each targeting a specific domain of network deployment, and such architectures could be classified as either it requires transmission of a minor update or a scripted application specification, or a full, binary application image for reprogramming. The former approach includes the Reijers[32] and Jeong[9] proposed method, which applies a platform independent patch to the target nodes in order to achieve reprogramming functionality. Mate[25] introduces a scripting approach, where the application script sent from the reprogrammer

would be interpreted and run on the end nodes using pre-build functions inside the end nodes. Though the updating or scripting method requires less packet transmission compared to full image propagation methods, the end sensor nodes are required to be smart and fat enough to comprehend and restructure the meta information sent by the reprogrammer. Another downside of this method, is that its capability to materialize applications are limited by its pre-build functions, which therefore lacks the flexibility to cope with on-the-fly user requests.

The full application code image transfer includes the works mentioned in [24], [35], [15], [20], [27], and [22]. A full binary image transfer requires higher bandwidth and longer code propagation time compared to scripting and updated methods. Under this restriction, Deluge[15] and MNP[20] enables partitioning of application code image into pages, and enables data transmission in a pipelining manner which enables efficient code propagation throughout the network. On the other hand, Deluge and MNP envisions reprogramming of the whole network, and does not assume reprogramming of a specified node.

**Deluge T2 Architecture**

Our code propagation mechanism derives from the Deluge architecture. Deluge is an application reprogrammable framework which runs on the TinyOS software stack which performs reprogramming of an entire sensor network by propagation of the application's binary execution image. The propagation of application code is done via trickle[25], and epidemic flooding protocol, throughout the whole network. For example, if an application were to be running on a sensor network and the user wishes to update the code image within the network, the sink node broadcasts an advertisement message for new code propagation. If the node which receives the broadcast message were to be running old code image compared to the advertised message, the target node requests an code update query to the advertiser, which then starts the transmission of application code image. The newly updated code

would then advertise the new code image it has just received, and in the same manner updates the outdated sensor nodes.

## 4.4 Core Mechanism

In this section we discuss the core mechanism of our architecture, localization of nearest nodes, and the code propagation mechanism to end nodes.

### 4.4.1 Node Localization

The localization algorithm applied for our architecture relies on RSSI values received by the propagation node by the sensing nodes. As described in Figure 4.1, the user query for application propagation is led by a broadcast message which demands an acknowledgment message from the sensing node. The message format of the request and acknowledgment message is described in Figure 4.3(b) and (c).

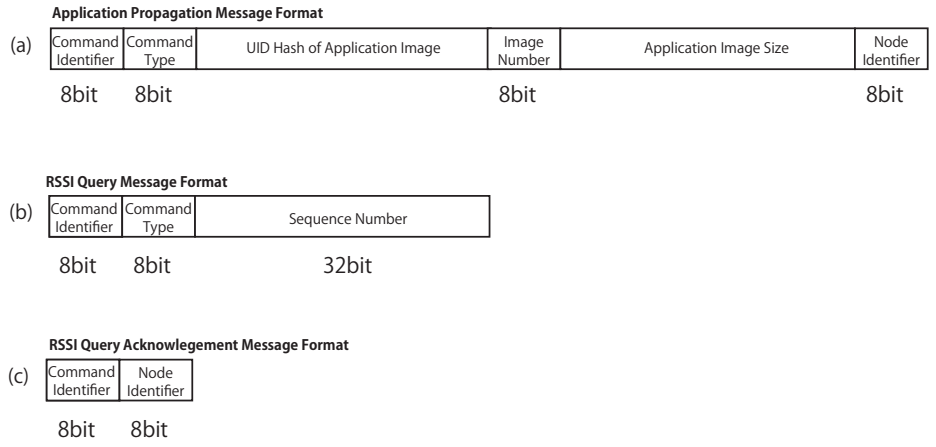


Figure 4.3: RSSI Query and Acknowlegement Message Format

The received RSSI signal is queued in a queue of size N in the propagation node, in which N is a function of the number of sensor nodes, ns to propagate code corresponding to a query. The queue is then linearly scanned for the largest RSSI values, and the sensor node ID corresponding to the ns largest

values are selected as the node to propagate code. Figure 4.4 illustrates our algorithm in detail.

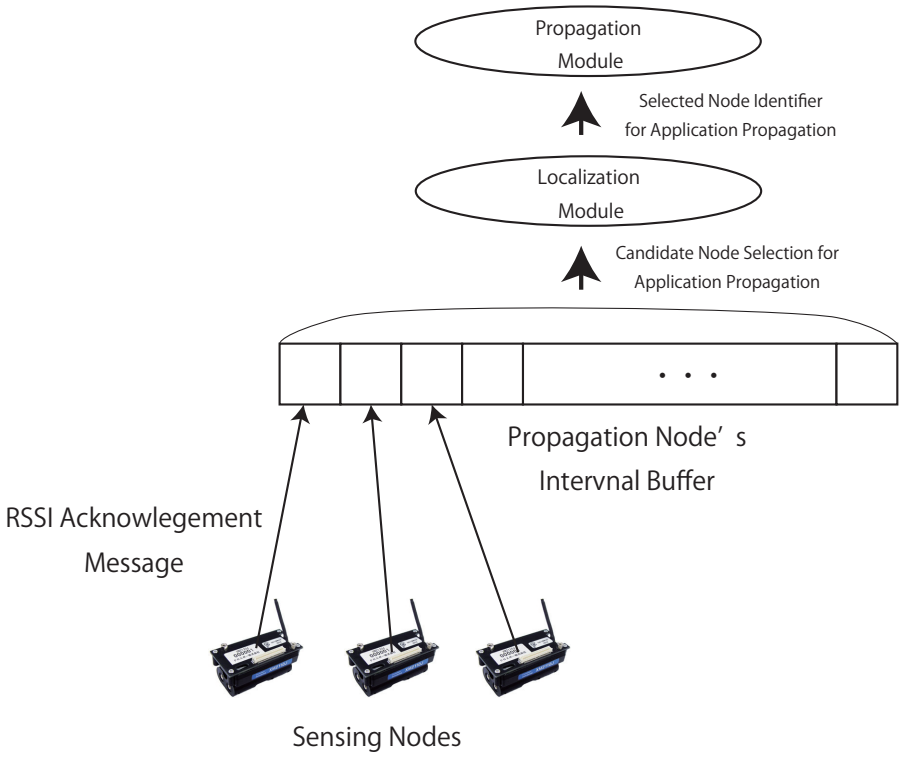


Figure 4.4: Node Localization Algorithm

### 4.4.2 Code Propagation

The requirements for application code propagation are, a) the capability for dynamic reprogramming over the radio, and b) the capability to support multiple applications. Several methods of over-the-air dynamic reprogramming of sensor network reprogramming exists. In our research, we base our code propagation mechanism on the Deluge, a dynamic reprogramming environment implemented on TinyOS. Below we will discuss the Deluge architecture and other dynamic reprogramming methods in a sensor network. Sending large objects in the Deluge architecture is facilitated via paging of the application code image. Among code transmission, the sender splits the

code image into manageable page size, which is then orderly transferred to the reception node. This allows the reception node to become the sender of the portion of the newly received code image, even if it has not finished completed downloading the whole code image.

# Chapter 5

# Evaluation

This chapter discusses the evaluation results of our work. We have performed our evaluation under a real-life testbed environment using 18 IRIS sensor nodes. We will first describe our experimental background, and then present the node localization and application code propagation time under our testbed. We then observe the experimental results and discuss our application for existing and future PCSNs.

## 5.1 Environmental Description

We have evaluated our work under a real-life environment using 18 IRIS sensor nodes. Our experiment took place in the central courtyard of the Keio Shonan Fujisawa Campus (62m x 18m) from 11:44, 12/29/10 through 0:28, 12/30/10. Environmental obstacles are shown in figure 5.1. Also, during the time span while the experiment was performed, no human movement were observed.

## 5.2 Hardware Implementation

We have implemented our mechanism on the IRIS platform, under the TinyOS software stack using the nesC language for development. The underlying functionality of our mechanism relies on the Deluge architecture.

Figure 5.1: Testbed Environment

Table 5.1 describes our implementation details and contribution in addition to the Deluge architecture.

Table 5.1: Implementation Details

| | |
|---|---|
| Software Development Environment | TinyOS 2.1.1 |
| Development Language | nesC 1.3.2[13] |
| Hardware Platform | IRIS Mote[37] |
| Line Count Comparison with Deluge T2 | +1628 lines |
| Application Image Size Comparison with Deluge T2 (ROM size) | +6828bytes |
| Application Image Size Comparison with Deluge T2 (RAM size) | +67bytes |

### 5.2.1 Preliminary Experiments

Localization and code propagation to the nearest sensor node is a key factor for success for our proposed architecture. In this section, we describe two experiments preformed in prior to our system's evaluation to understand the factors related to localization in our environmental field; the correlation of RSSI value to distance between two IRIS sensor nodes, and the code propagation completion time to distance among two nodes.

### 5.2.2 RSSI Value to Distance

We have measured the correlation of RSSI value and distance using two sensor nodes. Our experiment took place within point point a and point b in Figure 5.4. A single node which acts as the data sender would send 200 packets on a 1 second interval to the data collection node, which would retrieve the RSSI value of the sent packet. As mentioned in the previous chapter, RSSI values depend on the node's underlying radio transceiver. In our case we use the IRIS mote hardware platform which embed the AT86RF230 transceiver; therefore the RSSI value range from 0 (weak signal) to 28 (strong signal). Figure 5.2 describe the results of our experiment.

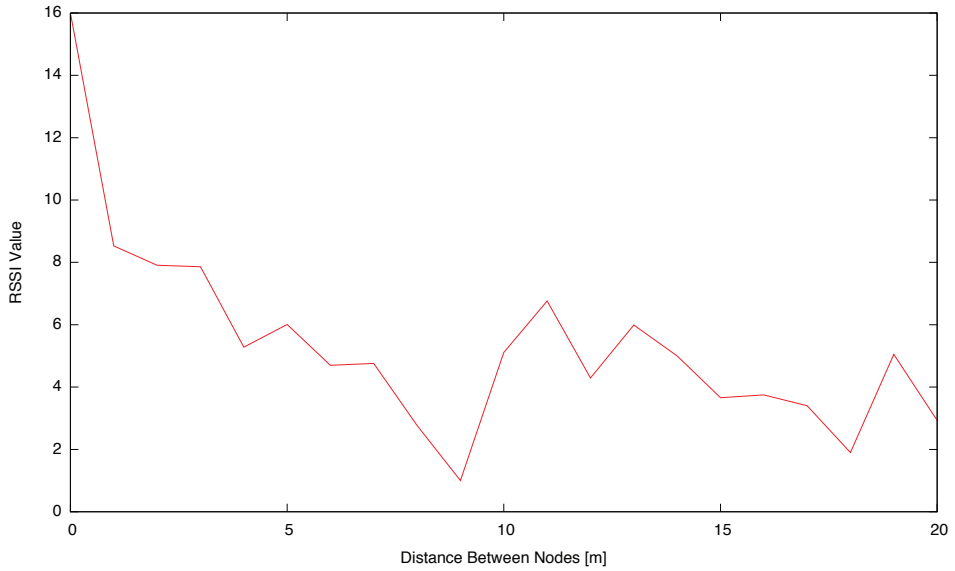


Figure 5.2: RSSI Value to Distance

The x-axis indicate the distance between the two nodes, while the y-axis represents the average RSSI value of 200 packets sent and the variance of the values. Though we could observe a moderate amount of variance between 6m to 9m, we could assume an exponential reduction in RSSI value as the nodes draw apart.

### 5.2.3 Code Propagation Completion Time to Distance

As the prior experiment, we have measured the application code propagation completion time to distance between the same two nodes. In our architecture, application code is propagated after nearest sensor nodes are localized. Therefore, we have defined code propagation completion time as the time frame within when the user have first issued the code propagation request to when the user have received the first application packet from the sensor node. Application code used for propagation is a simple sense and send application of 80KB in size, which broadcasts the temperature value to near nodes after node reboot. Application image is uncached on each try, therefore each trial requires a full code image transmission. Figure 5.3

describes the results of this experiment.

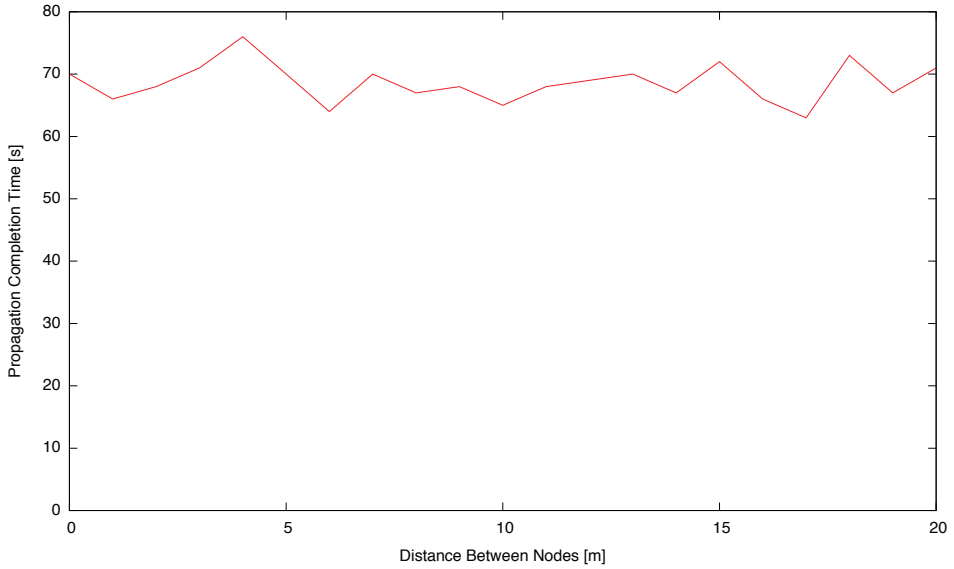


Figure 5.3: Code Propagation Completion Time to Distance

In contrast to the prior experiment, application code propagation time does not show correlation between distance, but instead uniform among all distances in our environment. The results of this experiment indicates that users could predict the code propagation time under various sensor network deployment density.

## 5.3 Evaluation Method

Figure 5.4 describes the sensor node placement in our testbed. 18 IRIS sensor nodes were consistently placed within the environment, simulating a grid-like sensor network deployment topology.

In this experiment, we simulate an application to sense the environmental data of the user's surroundings. The application assumes uses to interact with his handheld device for application propagation and data collection. At the time of the experiment, we have used an IRIS mote connected to a MacBook Pro via usb to serial programming board (5.5).

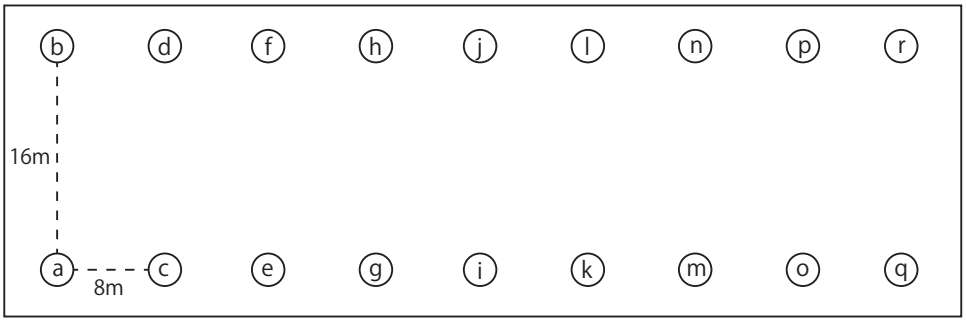


Figure 5.4: Deployment of our Testbed

Figure 5.5: Application Propagation Node

In this experiment, we evaluate the system-wide performance corresponding to the user traversal within the PCSN. The user first starts off from point a, in which he propagates a single application to two of his nearby nodes. The propagation of the application is notified to the user via application packet which is sent after reboot of the propagated sensor node. After reception of the two application packets, the user shifts to point b, where he performs the same operation as prior. The same application image as the preliminary experiment is used for propagation, and application image is cached on the sensor node's flash storage. Therefore, receiving application query which has already been submitted in the past would result the node to fetch code from his internal storage, rather than code transmission from the user's node. The user would traverse the field as marked in figure 5.6, until reaches the edge of the network.

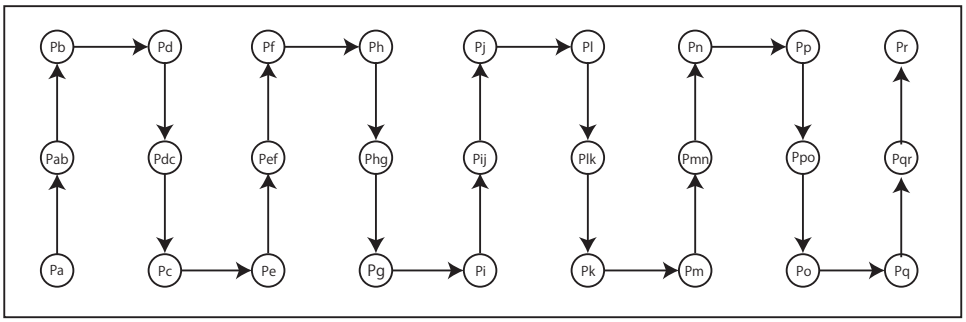


Figure 5.6: User Traversal Path

## 5.4 Results

From our experimental results, we have evaluated 3 system specific parameters of our method; response time of localization query, localization precision and code propagation time. In this section, we observe and analyze the results corresponding to each parameter.

### 5.4.1 RSSI Query Response Time

Figure 5.7 describes the RSSI querying overhead for our architecture. The x-axis represent the trial in which code propagation was performed, while the y-axis represent the RSSI querying overhead which is the time interval between RSSI querying from the user to propagation node specification within the propagation node. In our architecture, each deployed sensor node

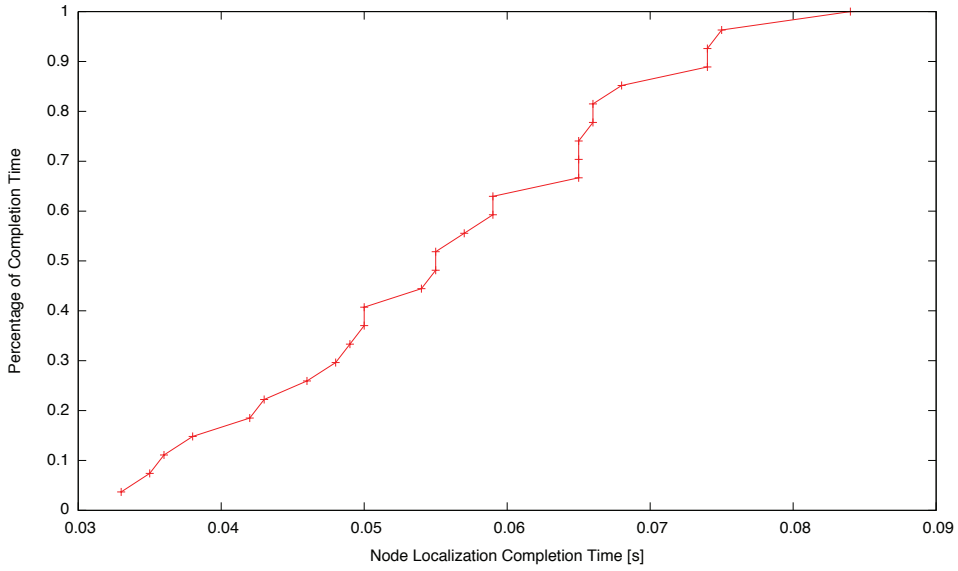


Figure 5.7: RSSI Query Response Time

reflexively sends a 2 byte packet consisting of the sender's node identifier and the command identifier on reception of the query command, and calculation of RSSI is done on the propagation node. Therefore, the overhead resides through 3 to 9 milliseconds, which combined with the preceding result of application code propagation overhead, negligible in most cases.

### 5.4.2 Node Localization Precision

In our testbed environment, the nodes are deployed in an uniform manner; i.e., lengthwise and crosswise, the nodes are apart from each other in a multiple of 8m. Now let us assume the minimal common multiple of distance between adjacent nodes, 8m, as a single unit of measurement, 1u. Under

this assumption, node a and c would be 1u apart, whereas 2u for nodes a and b. Evaluation of the precision of our localization method is performed as follows. At each measurement point, two nodes are chosen for application code propagation. The distance from the measurement point and the target nodes are triangulated using our theoretical measurement unit, and the mean value of distance for the two nodes are plotted in figure 5.8. The x-axis indicate the mean distance based on the theoretical unit, while the y-axis indicate the distribution function of the x-axis. The results indicate the localization range variance between 0u and 3.5u (0m to 28m), where under most cases, the distance between the propagation and sensing node resides below 2u (16m), while the average of all localization is 1.6u (12.8m).

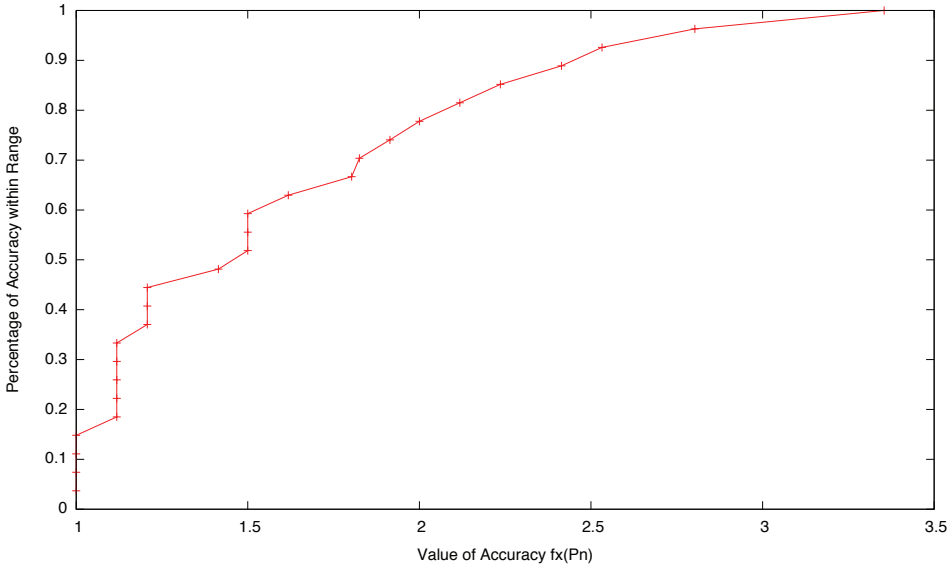


Figure 5.8: Node Localization Precision

### 5.4.3 Code Propagation Time

Figure 5.9 show the application code propagation time for our experiment. The x-axis represents the application code propagation time overhead which is the time interval after the propagation node has issued the application propagation command to when the propagation node has received the appli-

cation message from both nodes, while the y-axis represents the cumulative distribution of the x-axis values.

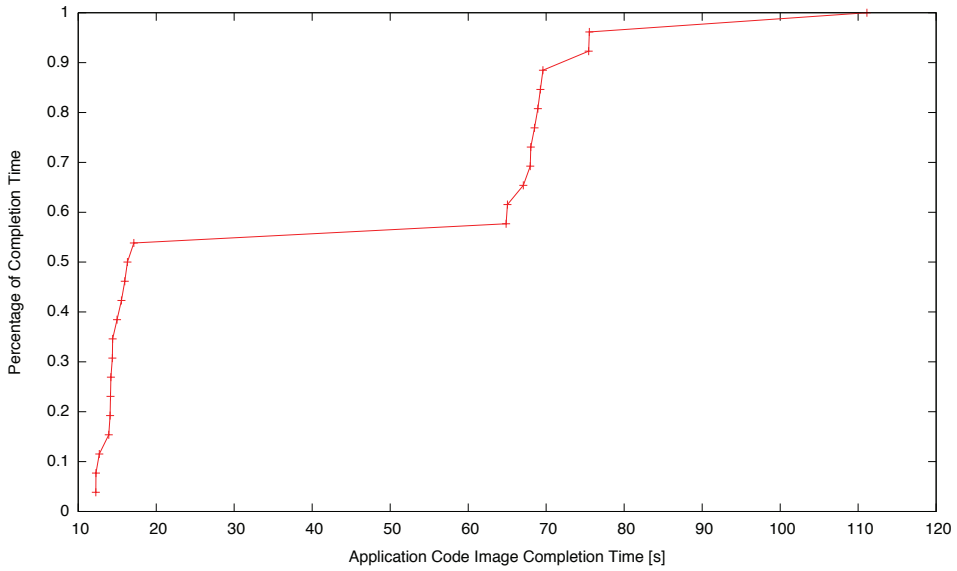


Figure 5.9: Code Propagation Time

From the results we could observe two clusters of data which represents respectively, a fast served user query and a slow served query. The former cluster represents the application message after code propagation from sensor nodes which have already received the same code image in the past, while the latter represents code propagation to the sensor nodes which have been newly selected as the candidate node for code propagation. If limited to code propagation completion for a single node, the distribution is as Figure 5.10.

## 5.5 Observations

Throughout our experiment, we have evaluated the time required for node localization, node localization precision, and application propagation time for the localized sensor nodes. The localization related results have been sufficient enough considering the deployment density of sensor nodes and

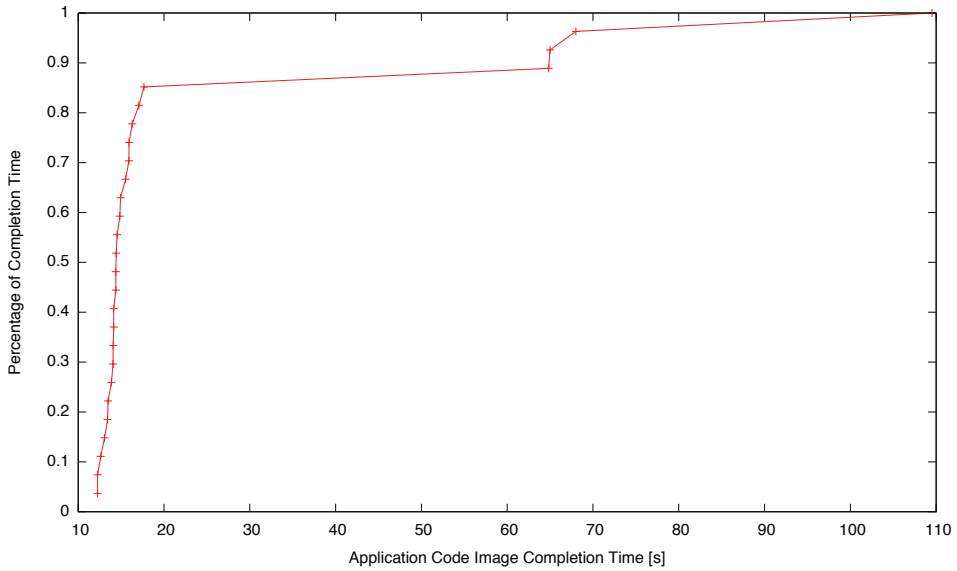


Figure 5.10: Code Propagation Time for a Single Node

deployment area. However, the initial code propagation completion time, although our work relies on the Deluge T2 framework, could be issued for further consideration; i.e., assuming an average person traversal velocity is 3km/h (50m/minute), the user could traverse far more than our minimal unit of traversal (1u), during the initial application propagation time. That being said, we list our future works to be done in the following chapter.

# Chapter 6

# Conclusion

In this chapter, we summarize our work and provide future research topics corresponding to our proposal.

## 6.1 summary

In our work, we have proposed a user driven code propagation architecture for human centric sensor networks. Our problem statement was that previous human centric network architectures lacks performance when the user's application requirement towards the sensor network is to sense his surroundings. The core design principle of our work is the direct user interaction (i.e., nearest node localization and direct code propagation) with the sensor network, which in our assumed environment could be done rapidly without redundancy, compared to a centralized approach. We have implemented our architecture on TinyOS, and have evaluated our work on a real-life testbed using 18 sensor nodes. Our evaluational results provide insights for future PCSNs that localization of sensor nodes could be done rapidly with succinct precision. Also under scenarios where the user traverses the same path more than once, the binary code image transmission overhead could be greatly reduced via caching, which has shed light to deployment of our architecture among such environments.

## 6.2 Future Works

Throughout our thesis, we have only focused on the architecture within the edge network. As our proposed architecture resides in the human centric sensing architectures mentioned in previous researches, we would need to consider the incorporation mechanics of our architecture with existing architectures. We have also limited the use-case to interaction between a single user and the network. In a real world environment, multiple users could reside within the same sensor network, which rises several research issues, such as resource sharing, privacy and security.

# Acknowledgements

First and foremost, I would like to express my deepest gratitude to my professor Hideyuki Tokuda for supervising me throughout my years in bachelors and masters degree. Not only have you have you generously tolerated my stupidity, but you have always brought me up when I was lost in academics, research and my life. I will always keep in my how you have literally saved my life in the fall of 2009 (details omitted), and promise my gratitude after graduation.

Members of the lab, especially fellows of ECN, I am who I am because I have spent my years surrounded by your outstanding minds and presence. Thank you very much.

Finally, I would like to thank my family for the continuous financial support and tolerance. I would not have been able to write fulfill my duties as a student without your everyday help.

# Bibliography

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, and M. Gouda. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *The International Journal of Computer and Telecommunications Networking*, 46(5):605–634, Dec. 2004.

[2] A. Baggio. Wireless sensor networks in precision agriculture. In *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005), Stockholm, Sweden*. Citeseer, 2005.

[3] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proceedings IEEE INFOCOM 2000 Conference on Computer Communications Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies Cat No00CH37064*, volume 2, pages 775–784. Ieee, 2000.

[4] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms. *Mobile Networks and Applications*, 10(4):563–579, 2005.

[5] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, N. Gordon, T. Bessell, M. Rutten, and S. Jha. Wireless sensor networks for battlefield surveillance. In *Proceedings of the Land Warfare Conference*, number October. Citeseer, 2006.

[6] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: sensor networks in agricultural production. *IEEE Pervasive computing*, pages 38–45, 2004.

[7] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. a. Peterson. People-centric urban sensing. *Proceedings of the 2nd annual international workshop on Wireless internet - WICON '06*, pages 18–es, 2006.

[8] J. Cui, J. Kong, M. Gerla, and S. Zhou. The challenges of building scalable mobile underwater wireless sensor networks for aquatic applications. *IEEE NETWORK*, 20(3):12, 2006.

[9] D. Culler. Incremental network programming for wireless sensors. *The First IEEE International Conference on Sensor and Ad hoc Communications and Networks*, pages 25–33, 2004.

[10] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks*, volume 0, pages 455–462. IEEE Computer Society, 2004.

[11] A. Eswaran, A. Rowe, and R. Rajkumar. Nano-RK: An Energy-Aware Resource-Centric RTOS for Sensor Networks. *26th IEEE International RealTime Systems Symposium RTSS05*, pages 256–265, 2005.

[12] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language. *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation PLDI 03*, page 1, 2003.

[13] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language. *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation PLDI 03*, page 1, 2003.

[14] L. Hu and D. Evans. Localization for mobile sensor networks. *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom '04*, (October):45, 2004.

[15] J. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94. ACM, 2004.

[16] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.

[17] A. Kansal, S. Nath, J. Liu, and F. Zhao. Senseweb: An infrastructure for shared sensing. *IEEE MultiMedia*, 14(4):8–13, 2007.

[18] B. Karp and H. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, volume pages, pages 243–254. AFOSR MURI Grant F49620-97-1-0382, and NSF Grant CDA-94-0124, and in part by Microsoft Research, Nortel, Sprint, ISI, and ACIRI, ACM, 2000.

[19] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks. *Wireless Networks*, 8(2/3):169–185, 2002.

[20] S. S. Kulkarni and L. Wang. MNP: Multihop Network Reprogramming Service for Sensor Networks. In *Proceedings of the 25th IEEE international Conference on Distributed Computing Systems ICSCS*, pages 7–16. Ieee, 2005.

[21] K. H. Kwong, T. T. Wu, H. G. Goh, B. Stephen, M. Gilroy, C. Michie, and I. Andonovic. Wireless Sensor Networks in Agriculture: Cattle Monitoring for Farming Industries. *Progress In Electromagnetics Research Symposium*, 5(1):31–35, 2009.

[22] P. Levis and D. Culler. The firecracker protocol. *Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC - EW11*, page 3, 2004.

[23] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and Others. Tinyos: An operating system for sensor networks. *Ambient Intelligence*, pages 115–148, 2005.

[24] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for sensor networks. *Ambient Intelligence*, 35:115–148, 2005.

[25] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle : A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. *In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation*, pages 15—-28, 2004.

[26] R. Murty, A. Gosain, M. Tierney, A. Brody, A. Fahad, J. Bers, and M. Welsh. CitySense: A vision for an urban-scale wireless networking testbed. In *Proceedings of the 2008 IEEE International Conference on Technologies for Homeland Security, Waltham, MA*. Citeseer, 2008.

[27] V. Naik, A. Arora, and P. Sinha. Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Wireless Embedded Devices. *26th IEEE International Real-Time Systems Symposium (RTSS'05)*, pages 277–286, 2005.

[28] D. Niculescu and B. Nath. Ad Hoc Positioning System ( APS ). In *Review Literature And Arts Of The Americas*, volume 5, pages 1–4. Ieee, 2001.

[29] N. Patwari and A. O. H. Iii. Using Proximity and Quantized RSS for Sensor Localization in Wireless Networks Categories and Subject Descriptors. *WSNA '03 Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, 2003.

[30] J. Postel. RFC 793: Transmission Control Protocol, 1981.

[31] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*, 2000(August):32–43, 2000.

[32] N. Reijers and K. Langendoen. Efficient code distribution in wireless sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, WSNA '03, pages 60–67. ACM New York, NY, USA, ACM Press, 2003.

[33] A. Savvides, C. Han, and M. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179. ACM, 2001.

[34] G. Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. *Proceedings of the 2nd international conference on Embedded networked sensor systems - SenSys '04*, page 1, 2004.

[35] T. Stathopoulos, J. Heidemann, D. Estrin, and C. U. L. A. C. F. E. N. SENSING. A Remote Code Update Mechanism for Wireless Sensor Networks, 2003.

[36] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin. PermaSense: investigating permafrost with a WSN in the Swiss Alps. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 8–12. ACM, 2007.

[37] C. Technology. IRIS Mote Specification Sheet, 2011.

[38] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, (April):18–25, 2006.

[39] C.-D. Wu, F. Chen, P. Ji, and Y.-Z. Zhang. A LEACH-based routing protocol for wireless sensor network to optimize QoS. *Dongbei Daxue XuebaoJournal of Northeastern University*, 30(8):1091–1094, 2009.

[40] D. Yang, X. Li, R. Sawhney, and X. Wang. Geographic and energy-aware routing in Wireless Sensor Networks. *International Journal of Ad Hoc and Ubiquitous Computing*, 4(2):61, 2009.

[41] F. Ye, G. Zhong, S. Lu, and L. Zhang. GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks. *Wireless Networks*, 11(3):285–298, 2005.