RFIDcover - A Coverage Planning Tool for RFID Networks with Mobile Readers

Dissertation

submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

by

S. Anusha

(Roll no. 03329033)

under the guidance of

Prof. Sridhar Iyer

Kanwal Rekhi School of Information Technology

Indian Institute of Technology Bombay

2005

Dedicated to my family

Dissertation Approval Sheet

This is to certify that the dissertation entitled

RFIDcover **- A Coverage Planning Tool for an RFID Network with Mobile Readers**

by

S. Anusha

(Roll no. 03329033)

is approved for the degree of **Master of Technology**.

____

Prof. Sridhar Iyer

(Supervisor)

____

Prof. Anirudha Sahoo

(Internal Examiner)

____

Dr. Pravin Bhagwat

(External Examiner)

____

Prof. Abhay Karandikar

(Chairperson)

Date: ______________

Place: ______________

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

CERTIFICATE OF COURSE WORK

This is to certify that **Ms. S. Anusha** was admitted to the candidacy of the M.Tech. Degree on 19th July 2003, and has successfully completed all the courses required for the M.Tech. Programme. The details of the course work done are given below.

Sr.No.	Course No.	Course Name	Credits
		Semester 1 (Jul – Nov 2003)	
1.	IT601	Mobile Computing	6
2.	IT603	Data Base Management Systems	6
3.	IT619	IT Foundation Laboratory	10
4.	IT621	Foundation course of IT - Part I	3
5.	IT623	Foundation course of IT - Part II	3
6.	IT694	Seminar	4
7.	HSS699	Communication and Presentation Skills (P/NP)	4
		Semester 2 (Jan – Apr 2004)	
8.	CS604	Combinatorics	6
9.	IT610	Quality of Service in Networks	6
10.	CS628	Introduction to Asynchronous Systems	6
11.	IT680	Systems Laboratory	6
		Semester 3 (Jul – Nov 2004)	
13.	CS681	Performance Analysis of Computer Systems	6
14.	HS617	Creating and Managing Intellectual Property Rights	6
		M.Tech. Project	
15.	IT696	M.Tech. Project Stage - I (Jul 2004)	18
16.	IT697	M.Tech. Project Stage - II (Jan 2005)	30
17.	IT698	M.Tech. Project Stage - III (Jul 2005)	42

I.I.T. Bombay   Dy. Registrar(Academic)

Dated:

Abstract

Radio Frequency Identification (RFID) finds use in numerous applications involving item identification and tracking. In a typical application, RFID tags are attached to the items and are periodically queried by readers. These applications require that a sufficient number of readers be deployed in order to provide complete coverage of all the tags in the given area.

Using a fixed placement of readers to guarantee complete coverage at all times increases the deployment costs. Also, most practical applications do not need complete coverage at all times. It is enough to provide complete coverage periodically, say each tag being covered every τ seconds. For such applications, using mobile readers to cover the area would be more cost-effective.

Given an area to be covered completely within a period τ, determining the number of mobile readers required, their placement and movement pattern, is a difficult problem. We have designed and developed *RFIDcover*, an automated coverage planning tool, that addresses this problem. Given an application scenario and reader specifications, *RFIDcover* determines an optimal number of readers required to guarantee complete coverage within the specified period τ. It also generates a layout giving the placement and movement pattern of the readers. In this thesis report, we present *RFIDcover* and its implementation for a retail inventory tracking application scenario and evaluate its effectiveness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Radio Frequency Identification System

Radio frequency identification, or RFID, is a generic term used for technologies that use radio waves to automatically identify individual items. The key components of an RFID system are the tags and the readers [1].

The RFID tag is a low functionality microchip with an antenna connected to it. The tag is attached to the item to be tracked, or identified, and stores the unique identification number of the item. They can be *active* or *passive*[2]. *Active* tags have their own battery power and use it for their on-tag computations as well as for communication; *passive* tags, on the other hand, derive the power for their computation and communication from the reader's signal.

The RFID readers are devices consisting of a transmitter/receiver module connected to an antenna. They communicate with the tags for reading/writing the information stored on them. The distance within which a reader can communicate with the tag is known as the *interrogation range* of the reader. Readers may be fixed (stationary at a location) or mobile (within the given area).

1.2 Applications

Radio Frequency Identification technology finds a plethora of applications in various commercial sectors. RFID can be used for tracking of objects like items in a supermarket, passenger baggage in the airlines industry, important documents in an office, animals in a farm or a national park, inventory items in a warehouse or manufacturing unit, and books in a library. It can also be used for identification at checkout counters, access control for buildings and parking lots, ticketing, pay-phone, and the like. RFID's basic advantage over other identification techniques is the full automation of the data capture, the ability to work from a greater distance without any need for the line of sight communication, and the ability to write onto the tag.

1.3 Problem Definition

RFID networks are constrained by the computationally thin, and limited power tags. For the system to be cost effective, the tags used for most applications are *passive*; this puts a limit on the *interrogation range* of the reader. For example, RFID readers operating in the UHF band typically have an interrogation range of 3-5 m[1]. As a result, a large number of readers may be required to provide complete coverage for a given area, leading to significant deployment costs.

Some applications, such as retail inventory tracking, need complete coverage only periodically, say each tag being covered every τ seconds. For such applications, using mobile readers to cover the area would be more cost-effective. However, before deploying the readers, it is necessary to answer many important questions, such as:

- how many readers are needed for providing complete coverage,
- where should the readers be placed,
- how should the mobile readers move and with what velocity,
- how does the number of readers required vary with increase or decrease in τ.

Thus, given an area to be covered completely within a period τ, determining an optimal number of mobile readers required, their placement and movement pattern, is a difficult problem.

1.4 Solution Outline

We have designed and developed *RFIDcover*, an automated coverage planning tool that addresses this problem. Given an application scenario and reader specifications, *RFIDcover* determines an optimal number of readers required to guarantee complete coverage of an area within the specified period τ. It does this by:

(i) automatically generating a set of layouts (the placement and movement pattern of the readers),

(ii) computing the performance metrics - *Cost of deployment* (as a function of the number of readers) and *Tag Reading Time (TRT)* (the time taken to read all tags in the given area) - for each layout, and

(iii) selecting the layout which is optimal in terms of both.

The architecture of *RFIDcover* (Chapter 3) is generic and extendible, making it easy to implement different application scenarios. We consider the following retail inventory tracking application: An RFID tag is attached to each item in a supermarket. The items are then stacked up on the shelves separated by aisles. Periodic inventory checks are carried out using mobile readers moving along the aisles, reading the tags on the shelves as they move. We have implemented *RFIDcover* for such an application (Chapter 4), and evaluated its effectiveness (Chapter 5).

To the best of our knowledge, there is no literature on coverage tools for RFID systems with mobile readers.

Chapter 2

The Coverage Problem

In this Chapter, we discuss the problem of providing complete coverage using fixed as well as mobile readers. We assume an interference free environment and a circular range for each reader. We derive theoretical results for the number and placement of readers to completely cover a given area. These results form the basis for the mobility models and heuristics used in *RFIDcover*.

2.1 Fixed Readers

Given the minimum-area rectangle (with dimensions $X \times Y$) that encloses the area to be covered, the problem of complete coverage using fixed readers is same as that of covering this rectangle with a number of fixed size circles, each of radius equal to the readers' *interrogation range*, r. Ideally, the minimum number of readers required should be simply

$$F_i = \frac{Area\ of\ the\ bounding\ rectangle}{Area\ of\ the\ circle} = \frac{XY}{\pi r^2} \tag{2.1}$$

However, using only F_i readers may not ensure complete coverage, since it implicitly assumes that no two circles overlap. The hexagonal layout shown in Figure 2.1 (a) leads to the maximal cover possible [3] using non-overlapping circles. The percentage of the area covered can be computed as follows.

Consider the equilateral triangle formed by joining the centers of three neighboring circles shown in Figure 2.1 (a). This triangle forms the basic unit for coverage. We see from the figure that

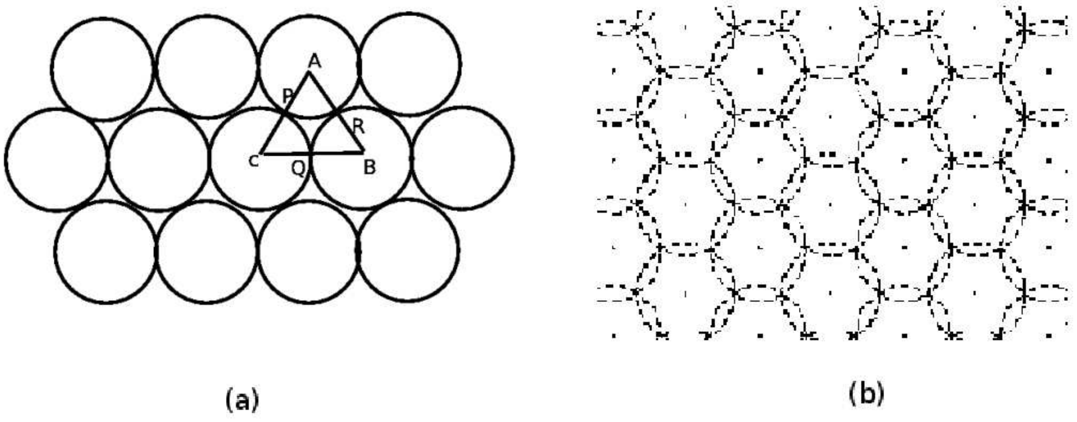


Figure 2.1: The Hexagonal Layout with Fixed Readers

$$\begin{aligned}
\%\ cover &= \frac{Ar(sector\ APR) + Ar(sector\ BQR) + Ar(sector\ CQP)}{Ar(trianlgeABC)} \\
&= \frac{3 \times \frac{\pi r^2}{6}}{\sqrt{3}r^2} \\
&= 0.906899682 \qquad (2.2)
\end{aligned}$$

Hence, complete coverage is possible only if there is overlap amongst the circles. Obviously, it would require more than F_i readers. Now, an optimal layout is the one that results in minimum overlap, i.e., uses minimum number of circles to completely cover the area. Kershner, in [4], states that the density of an optimal layout, i.e., the ratio of sum total of the area covered by all the circles to the total area to be covered, would be 1.209. Figure 2.1 (b) [5] shows a layout that meets this criteria, and hence, is optimal. The number of readers required for complete coverage in this case would be:

$$F_{opt} = \frac{2\sqrt{3}XY}{9r^2} \qquad (2.3)$$

It can be easily seen that using only fixed readers may not be cost-effective as X and Y increase, especially if $r \ll X$, and/or, Y. For example, to cover an area of dimension $10m \times 10m$ with readers of interrogation range, $r = 2m$, the number of readers required would be 10. Whereas, to cover an area of dimension $50m \times 50m$, this would shoot up to 241. Hence, we explore coverage using mobile readers.

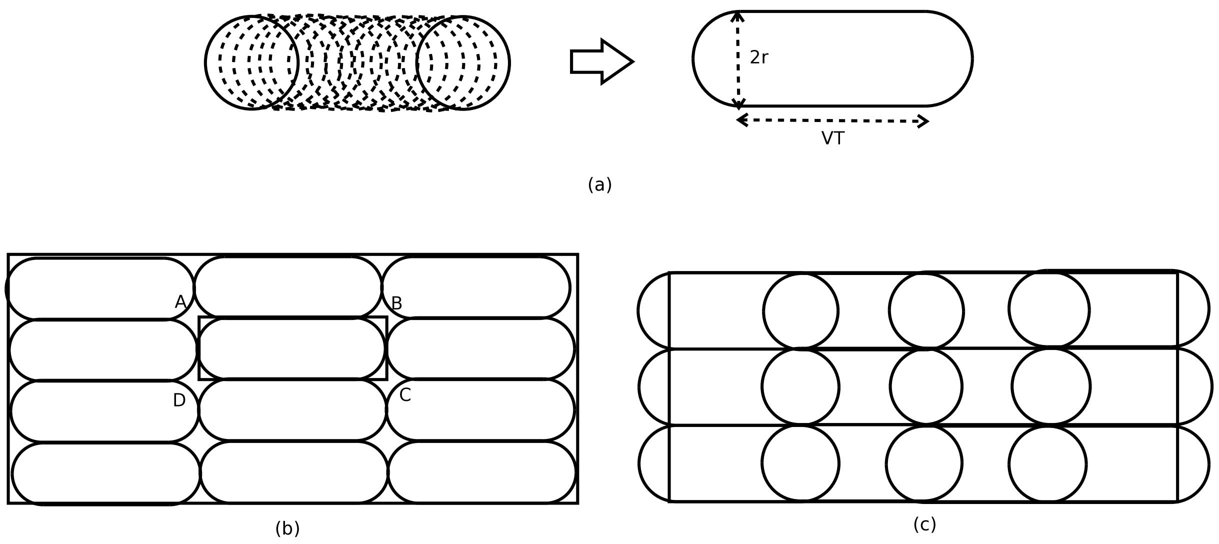


Figure 2.2: Coverage using Mobile Readers

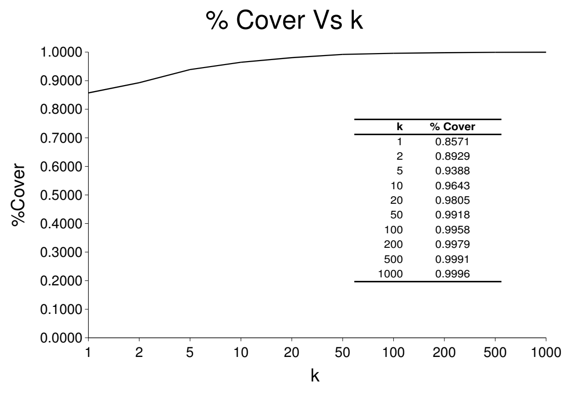
2.2 Mobile Readers

The approach for determining the optimal number of mobile readers to cover a given area is similar to that used for fixed readers, except for the following difference: The area covered by a mobile reader would not be of the shape of a circle of radius r, and would instead look as shown in Figure 2.2 (a). In the figure, r is the *interrogation range* of the mobile readers, v is the velocity with which the reader moves and τ is the time within which the area is to be completely covered.

The problem of complete coverage using mobile readers is now same as that of covering the minimum-area rectangle, with such ellipse-like shapes. Ideally, the minimum number of mobile readers needed should be:

$$M_i = \frac{Area\ of\ the\ bounding\ rectangle}{Area\ of\ the\ ellipse\ like\ shape} = \frac{XY}{(2v\tau + \pi r)r} \tag{2.4}$$

However, as in the case of fixed readers, this would not guarantee complete coverage, since there is no overlap assumed. In case of non-overlapping areas, we determine the $\%cover$ as follows: Consider the rectangle ABCD enclosing the ellipse-like shape as shown in Figure 2.2 (b). This rectangle forms the basic unit for coverage in this case. It can be seen from the figure that



k	% Cover
1	0.8571
2	0.8929
5	0.9388
10	0.9643
20	0.9805
50	0.9918
100	0.9958
200	0.9979
500	0.9991
1000	0.9996

Figure 2.3: % Cover Vs k

$$\begin{aligned} \%\ cover &= \frac{Ar(ellipse\ like\ shape)}{Ar(rectangleABCD)} \\ &= \frac{v\tau \times 2r + \pi r^2}{(v\tau + 2r) \times 2r} \end{aligned} \tag{2.5}$$

Let k be the largest integer such that $kr \leq v\tau$, i.e, $k = \lceil \frac{v\tau}{r} \rceil$. We get

$$\%\ cover = \frac{2k + \pi}{2k + 4} \tag{2.6}$$

and $\%cover$ varies with k as per the graph shown in Figure 2.3.

Thus, we see that complete coverage can be provided only if overlaps are allowed. One such layout with overlaps that provides complete coverage is shown in Figure 2.2 (c). The number of mobile readers required for complete coverage as per this layout is

$$M = \lceil \frac{X \times Y}{2rv\tau} \rceil \tag{2.7}$$

Although M may not be the *minimal* number of mobile readers required, it gives a *sufficient bound* for the number of mobile readers. We note that a deployment using M readers would need a mobility model for each reader that may be restrictive and impractical for many scenarios. Hence, we use this value of M only as a comparison point for the evaluation of mobility models implemented in *RFIDcover*.

In the next Chapter, we describe the architecture of *RFIDcover* in detail.

Chapter 3

RFIDcover Architecture

3.1 Three Phased Operation

The architecture of *RFIDcover* is as shown in Figure 3.1. It has a three phase operation as follows:

1. *Selection Phase* - In this phase, the mobility model for the mobile readers and the MAC mechanism to be used by the readers for shared access to the medium, are chosen based on the application scenario. Depending on these two, an appropriate heuristic for layout generation is selected.

2. *Generation Phase* - In this phase, the selected heuristic is used to generate a set of possible layouts, each of which conforms to the input constraints and also completely covers the given area. The performance metrics, viz., the *Cost* (as a function of the number of readers), and the *TRT* (total time taken to read all the tags in the entire area) are computed for each such layout.

3. *Optimization Phase.* In this phase, an appropriate objective function for optimization is chosen and applied to the set of layouts generated earlier. This results in the selection of an optimal layout, which is recommended to the user.

An additional feature of *RFIDcover* is that the user can also provide constraints on the *Cost*, *TRT* online as an input to the generation and optimization phases.

The following sections briefly describe the different components shown in the architecture and their roles, using the supermarket inventory application mentioned earlier (Section 1.4).

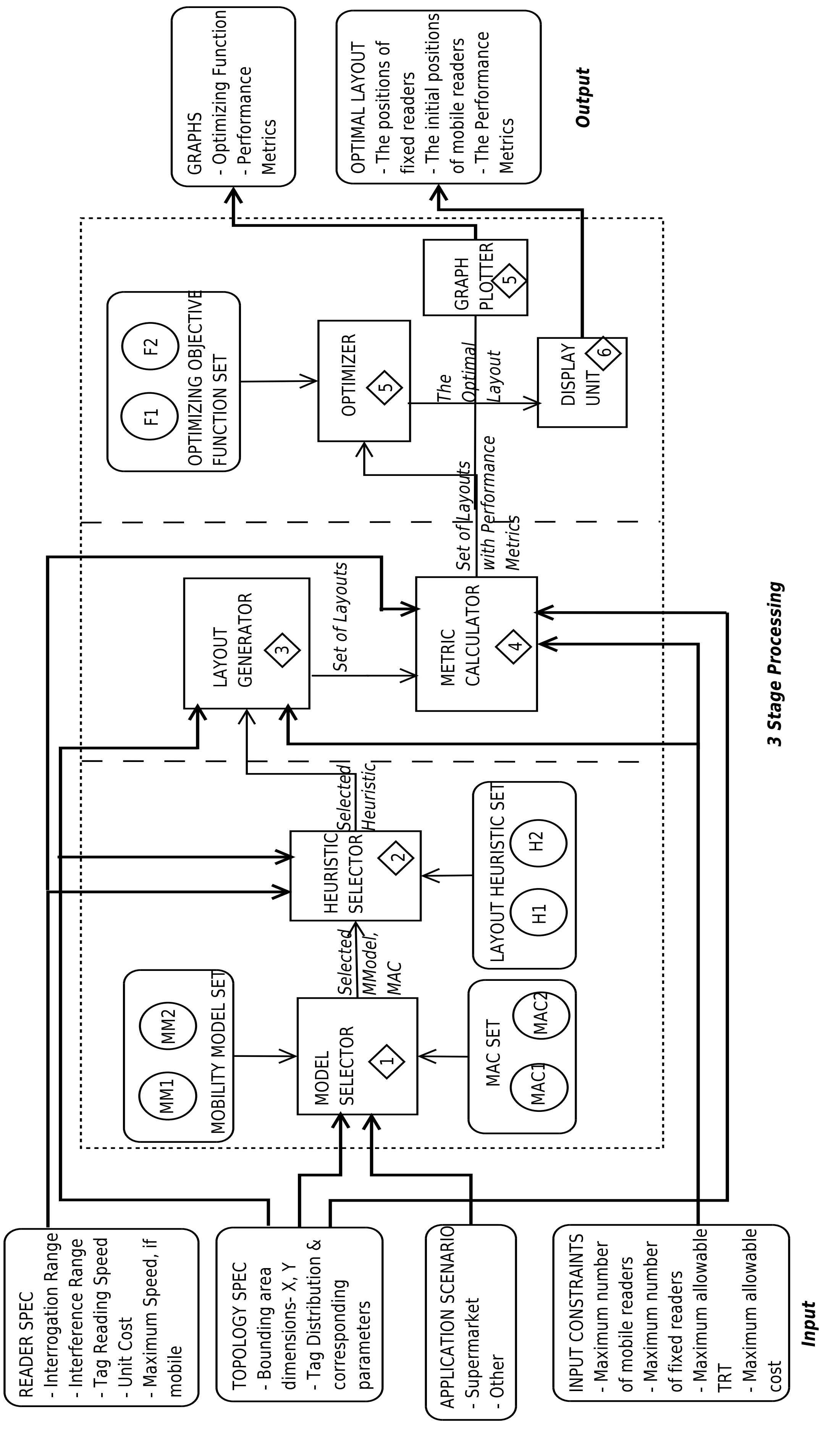


Figure 3.1: *RFIDcover* Architecture

3.2 Inputs

The user provides as input to *RFIDcover* the following:

- *Reader Specification* - This gives the details of both fixed and mobile readers. It includes the *interrogation range* - the distance upto which a tag, if present, can be read by the reader; the *interference range* - the distance within which if two readers transmit simultaneously their signals would interfere; the *Tag Reading Speed (TRS)* - the number of tags the reader can read in a unit time; and the *unit cost* of the reader. If the reader is mobile then the *maximum speed* with which it can move is also included.

- *Topology Specification* - This gives the details of the region to be covered. Whatever be the shape of the region, it is assumed that the user provides the *dimensions* of the minimum-area rectangle enclosing it. Throughout this thesis report, this rectangle is referred as the "area to be covered". Other than this, the topology specification also includes the *tag density distribution* and the corresponding *parameters.* For example, in the supermarket scenario tags are assumed to be uniformly distributed within each shelf and the tag density (tags per unit area) would be the corresponding parameter.

- *Application Scenario* - This is the application for which the RFID network is being deployed and for which *RFIDcover* is being used. The user chooses the application scenario from a list of those that are currently supported by *RFIDcover* and would include any application specific parameters. For our supermarket example, these parameters would be the *aisle length* and the *inter aisle distance.*

- *Input Constraints* - Additional constraints may be provided by the user, such as, the number of fixed and/or mobile readers to be used, the maximum allowable *Cost*, the maximum allowable *TRT (Tag Reading Time).* These constraints can be modified on-the-fly by the user, to generate a new set of layouts. This feature enables a novice user to start with some not-so-strict constraints and vary it online based on the output produced.

3.3 Functional Overview

Brief description of the functions of each of the basic components in *RFIDcover* is given below. The various components have been loosely grouped together to correspond to the three phases of its operation.

3.3.1 Selection Phase

Once the inputs are read they are used by the *selection phase* for selecting the appropriate mobility model, MAC mechanism and the layout generating heuristic. The two main components in this phase are:

- *Model Selector* : The model selector maps the given application scenario and tag distribution to a suitable mobility model and an appropriate MAC (medium access control) mechanism. It then passes on the selected mobility model and MAC mechanism as an output to the next phase. For our supermarket example, the *static coloring MAC* mechanism and the *zig-zag* mobility model are selected, as described in Chapter 4. The model selector uses the following components for the mapping:
  - *Mobility Model Set* : This is a collection of mobility models. A mobility model defines how the mobile readers would move and with what velocity. For example, one simple mobility model could be the *to-and-fro* model as follows: the mobile readers move from the left end to the right end of the area to be covered in a straight line, and then return back to the initial position through the same path. We assume a homogeneous system, where all the mobile readers follow the same mobility model. A given mobility model may be suitable for some application scenarios and may not be useful for others. Also, note that a mobile reader can not move faster than $TRS/(2r \times Tag\ density)$ in order to cover all tags in the range.
  - *MAC Set* : The readers may use a number of Medium Access Control (MAC) mechanisms, such as TDMA, FDMA, CSMA, in order to share the medium and read the tags efficiently. MAC set is the collection of such mechanisms currently supported by *RFIDcover*. Any new MAC mechanism can be implemented and added to this set.

- *Layout Generating Heuristic Selector* : Once the mobility model and MAC mechanism has been fixed, and given the tag distribution, reader specifications and the dimensions of the area to be covered, the layout generating heuristic selector finds an appropriate heuristic to be used for generating the layouts. These layouts provide complete coverage of the area and conform to the input constraints. For our supermarket example, the LGH_1 heuristic will be selected as it supports the *zig-zag* mobility model. The layout generating heuristic selector uses the following component for its selection decision:

  – *Layout Generating Heuristic Set* : This is a collection of heuristics that can be used for generating the layouts. For example, a layout generating heuristic could place fixed readers at four corners of the area to be covered and place one mobile reader at left top corner. Each heuristic has associated with it some mobility models that it supports. Hence, a heuristic along with the mobility model would result in a layout for complete coverage of the area.

3.3.2 Generation Phase

At the end of the *selection phase* we have all the information needed for the *generation phase*, for generating the layouts for complete coverage. This phase consists of two steps and uses the following components:

- *Layout Generator* : It takes as input the topology specifications, the reader specifications, the input constraints, and the chosen layout generating heuristic, and applies that heuristic to generate a set of possible layouts that would provide complete coverage in conformance with the input constraints.

- *Metric Calculator* : For each layout generated by the layout generator, the performance metrics are computed, and keeping in mind the input constraints, a subset of the layouts that conform to the input constraints is generated. The performance metrics include simple generic ones like the number of fixed and mobile readers, the total *Cost* incurred and the *TRT* (or Tag Reading Time, which is the total time it takes to read all tags in the area). In addition, other metrics specific to MAC mechanism or application scenario could also be computed.

3.3.3 Optimization Phase

Once the layouts conforming to the input constraints have been generated, the *optimization phase* then determines best layout from the set of those generated. The basic building blocks of this phase are:

- *Optimizer* : Since the layout is generated using a heuristic, there can be a number of layouts that would conform to the input constraints. Various optimizing objective functions are possible. The optimizer applies a suitable optimizing objective function to the layouts generated and recommends the result to the user, along with a summary of the other conforming layouts. The optimizer uses the following component while choosing the objective function.

  – *Optimizing Objective Function Set* : This is a collection of optimizing objective functions. For each layout generated that conforms to to the input constraints, we have a set of performance metrics. An optimizing objective function can be applied on some/all of these metrics for determining the most suitable layout. For our supermarket example, we might be interested in recommending a layout that uses the minimum number of readers. Then the objective function should be the *minimum* function and it should be applied on the *number of readers.*

- *Graph Plotter* : This is used to plot graphs using the layouts conforming to the constraints. The plots reflect the variation of number of fixed readers, mobile readers, *Cost*, *TRT* and other metrics with different layouts. It also displays the optimizing objective function.

- *Display Unit* : This is used to graphically display in detail the optimal layout and the performance metrics associated with it.

3.4 Outputs

Following are the outputs that *RFIDcover* provides to the user.

- *Graphs* : The various graphs generated by the graph plotter are shown to the user for analysis. This is especially useful since the user can tune the constraints online and get a layout more suitable to his constraints. The plotted points are also saved in a set of output files.

- *Optimal Layout* : The layout recommended to the user is shown graphically in detail. The details include the number of mobile readers and their initial positions, the number of fixed readers and their positions, the velocity of the mobile readers and their path of movement, the *Cost* of deployment and the *TRT*.

In the next Chapter, we present the implementation of *RFIDcover* for the retail inventory tracking application.

Chapter 4

RFIDcover Implementation

The architecture presented in Chapter 3 has been implemented in Java. The flexibility of the architecture is reflected in the design making it easy to support any application scenario. *RFIDcover* implementation currently supports the supermarket inventory application scenario. It implements the *zig-zag* mobility model and the *static coloring MAC* that are suitable for the supermarket application. LGH_1, a heuristic specific to the zig-zag mobility model is used to generate the set of layouts. The *least square sum* optimizing function is then applied on the parameters *TRT* and *Cost*, to get the optimal layout that is suggested to the user.

4.1 *RFIDcover* Design

The class diagram depicting the high level design of *RFIDcover* is as shown in Figure 4.1. As can be seen, the design is very close to the architecture described in Chapter 3, thus making the implementation of *RFIDcover* extendible to a large extent.

The `InputReader` reads all the input specifications and sets the corresponding value in the `InputObject`. It contains all the input specifications and also the models and heuristics selected later using these input specifications. The `GlobalRegistry` stores a list of applications, mobility models, layout generating heuristics and optimizing objective functions currently supported by *RFIDcover*. It also stores information regarding the corresponding applications or mobility models that they support. This registry is required to be updated whenever a new application, or mobility model, or MAC mechanism, or layout generating heuristic, is implemented and added to *RFIDcover* as an extension.

An appropriate implementation of the `ApplicationScenario`, `MobilityModel`, `MACMechanism`, `LGHeuristic` and `OptimizingObjectiveFunction` are instantiated at runtime

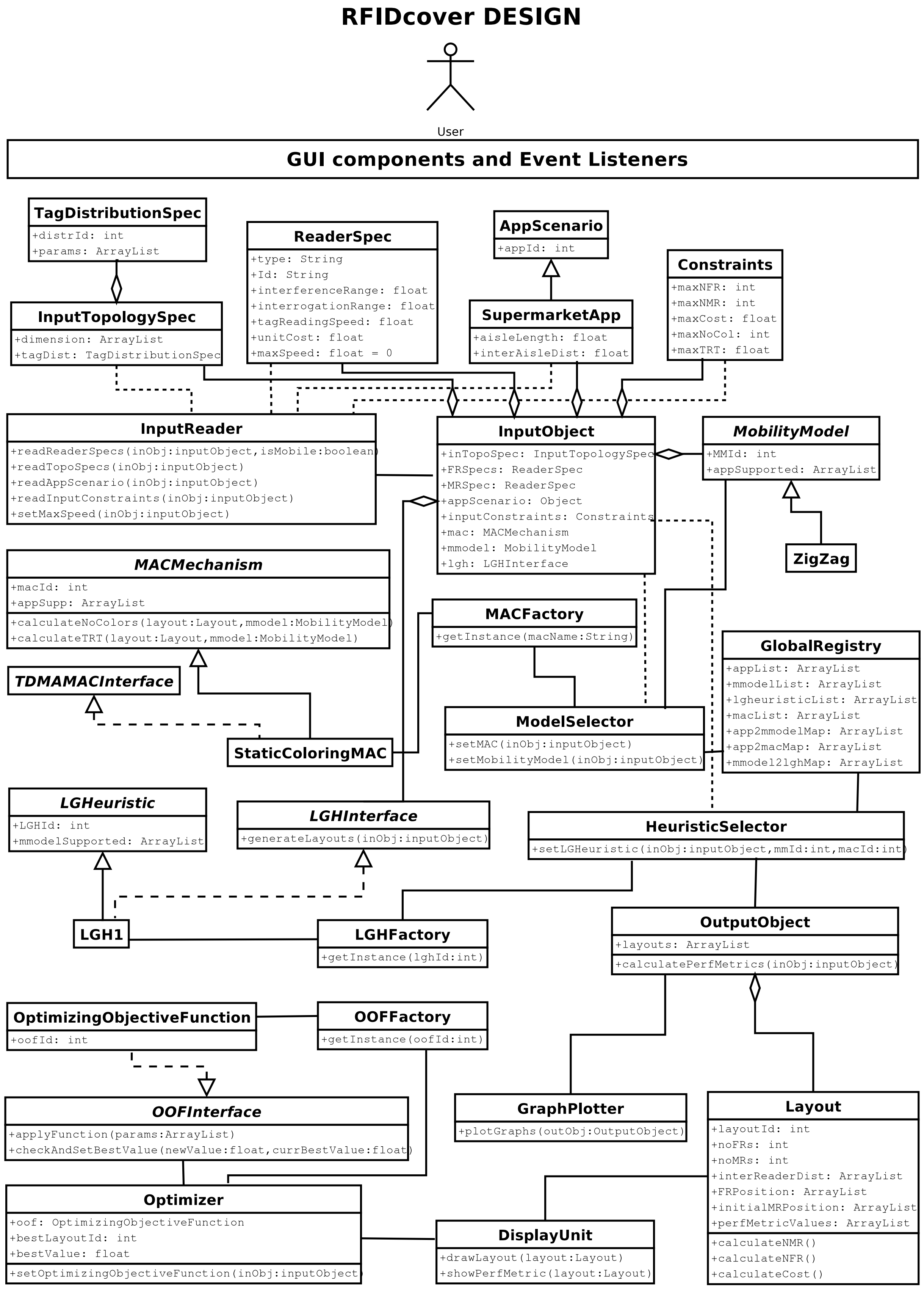

RFIDcover DESIGN
User
GUI components and Event Listeners
TagDistributionSpec
+distrId: int
+params: ArrayList
InputTopologySpec
+dimension: ArrayList
+tagDist: TagDistributionSpec
ReaderSpec
+type: String
+Id: String
+interferenceRange: float
+interrogationRange: float
+tagReadingSpeed: float
+unitCost: float
+maxSpeed: float = 0
AppScenario
+appId: int
SupermarketApp
+aisleLength: float
+interAisleDist: float
Constraints
+maxNFR: int
+maxNMR: int
+maxCost: float
+maxNoCol: int
+maxTRT: float
InputReader
+readReaderSpecs(inObj:inputObject,isMobile:boolean)
+readTopoSpecs(inObj:inputObject)
+readAppScenario(inObj:inputObject)
+readInputConstraints(inObj:inputObject)
+setMaxSpeed(inObj:inputObject)
InputObject
+inTopoSpec: InputTopologySpec
+FRSpecs: ReaderSpec
+MRSpec: ReaderSpec
+appScenario: Object
+inputConstraints: Constraints
+mac: MACMechanism
+mmodel: MobilityModel
+lgh: LGHInterface
MobilityModel
+MMId: int
+appSupported: ArrayList
ZigZag
MACMechanism
+macId: int
+appSupp: ArrayList
+calculateNoColors(layout:Layout,mmodel:MobilityModel)
+calculateTRT(layout:Layout,mmodel:MobilityModel)
TDMAMACInterface
StaticColoringMAC
MACFactory
+getInstance(macName:String)
ModelSelector
+setMAC(inObj:inputObject)
+setMobilityModel(inObj:inputObject)
GlobalRegistry
+appList: ArrayList
+mmodelList: ArrayList
+lgheuristicList: ArrayList
+macList: ArrayList
+app2mmodelMap: ArrayList
+app2macMap: ArrayList
+mmodel2lghMap: ArrayList
LGHeuristic
+LGHId: int
+mmodelSupported: ArrayList
LGHInterface
+generateLayouts(inObj:inputObject)
HeuristicSelector
+setLGHeuristic(inObj:inputObject,mmId:int,macId:int)
LGH1
LGHFactory
+getInstance(lghId:int)
OutputObject
+layouts: ArrayList
+calculatePerfMetrics(inObj:inputObject)
OptimizingObjectiveFunction
+oofId: int
OOFFactory
+getInstance(oofId:int)
OOFInterface
+applyFunction(params:ArrayList)
+checkAndSetBestValue(newValue:float,currBestValue:float)
GraphPlotter
+plotGraphs(outObj:OutputObject)
Layout
+layoutId: int
+noFRs: int
+noMRs: int
+interReaderDist: ArrayList
+FRPosition: ArrayList
+initialMRPosition: ArrayList
+perfMetricValues: ArrayList
+calculateNMR()
+calculateNFR()
+calculateCost()
Optimizer
+oof: OptimizingObjectiveFunction
+bestLayoutId: int
+bestValue: float
+setOptimizingObjectiveFunction(inObj:inputObject)
DisplayUnit
+drawLayout(layout:Layout)
+showPerfMetric(layout:Layout)

Figure 4.1: *RFIDcover* Design - The Class Diagram

based on what is selected in the *selection phase*. To facilitate this, we use the *object factory* design.

The `OutputObject` is a collection of layouts that are generated by the chosen `LGHeuristic` implementation. For each `Layout` in the `OutputObject` the performance metrics are computed and those which do not conform to the `Constraints` provided as input are removed from this collection. Some of these metrics are generic and computed directly from the layout, but others may be dependent on the MAC mechanism or mobility model and are computed accordingly. The conforming layouts are used by the `GraphPlotter` to plot and display the output graphs presented to the user. The `Optimizer` then applies the chosen `OptimizingObjectiveFunction` and sets the best layout, which is displayed in detail by the `DisplayUnit`.

As can be seen, extending *RFIDcover* is easily supported by this design. For example, if a new layout generating heuristic, is to be added, all we need to do is to implement a new class that extends the `LGHeuristic` class and implements the `generateLayouts` method appropriately, returning a set of layouts as a result. Also, we would need to first register the new heuristic in the `GlobalRegistry`

4.2 Zig-zag Mobility Model

The *zig-zag* mobility model is meant for a hybrid network, with a number of fixed and mobile readers. The mobile readers move within a rectangular area in a zig-zag fashion. This rectangular area (as shown in Figure 4.2) forms the basic unit which is replicated throughout the given area. So, the dimensions of this rectangle could be as large as $X \times Y$, or as small as *aisle length* $\times$ *inter aisle distance*.

This mobility model is suitable for supermarket scenario where tagged items are stacked up on shelves in rows with aisles separating them. The mobile readers move from left to right (or right to left) along the aisle and then move an *inter aisle distance* perpendicular to it, and then move from right to left (or left to right) along the aisle and repeat the whole process again. The model is as shown in Figure 4.2.

This mobility model works especially well if the time required to move in the perpendicular direction, that is the *inter aisle distance*, is very small in comparison to $v\tau$, where v is the velocity of the mobile readers and τ is the time within which the area must be

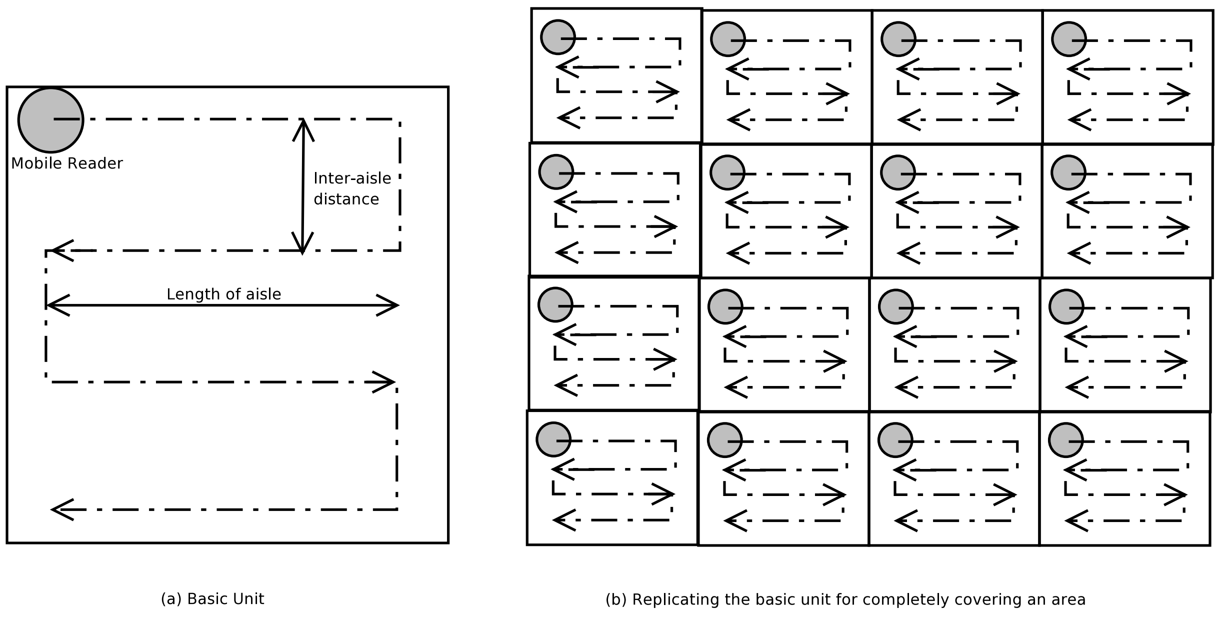

Mobile Reader
Inter-aisle distance
Length of aisle
(a) Basic Unit
(b) Replicating the basic unit for completely covering an area

Figure 4.2: The Zig-zag Mobility Model

completely covered.

4.3 Static Coloring MAC

The *static coloring MAC* is a TDMA based MAC mechanism for shared access of the readers to the medium of communication. We chose a TDMA based mechanism because this is simple and is commonly used by commercially available RFID readers. FDMA is not popular due to the overhead of frequency switch by antennas, and CDMA requires more computational power at the tags.

In a TDMA based mechanism, each reader is assigned a time slot wherein it can read tags in its range, while the rest of the readers remain silent during the period. Any two readers in the interference range of each other will be assigned different time slots, thus, avoiding any interference or collision. Assigning time slots to readers is analogous to the graph coloring problem, where the readers form the vertices of the graph and every pair of interfering readers are adjacent, that is, they have an edge between them.

In case of fixed readers, determining these pairs of interfering readers and assigning TDMA slots is relatively straightforward for specific topology. Engel et al present some centralized global graph-formulations in [6], for specific topologies that recur in RFID reader networks.

In case of mobile readers, determining the number of TDMA slots is not easy, since the readers may move in and out of the interference range of each other. One simple way to overcome this difficulty is to determine the worst case number of interfering readers and assign as many slots as needed. The *static coloring MAC* considers the mobility model, determines all pairs of readers that might come into the interference range of each other, and assigns time slots on a worst case basis.

This approach works for many simple mobility models and deterministic layouts, although in general it may be a bit inefficient. For the supermarket application, where the mobile readers follow the *zig-zag* mobility model, this MAC mechanism is used. In our application, if *inter aisle distance* is greater than the readers' interference range, the fixed readers do not interfere with other fixed readers. All the fixed readers, therefore, can read in the same slot. Each mobile reader may come into the range of the fixed readers at some point in time. Further, only two mobile readers can come into each other's interference range. So, all the mobile readers can take turns to read by using two slots amongst them. Hence, the *static coloring MAC* assigns three time-slots for the supermarket application.

4.4 LGH_1 Layout Generating Heuristic

RFIDcover currently implements a heuristic, which we call LGH_1, for generating the layouts appropriate for the *zig-zag* mobility model. It generates a hybrid layout consisting of fixed as well as mobile readers. The layouts differ in the strategic points where the fixed readers are placed.

LGH_1 works as follows: It places fixed readers at the end of every aisle, or at the end of every two aisles and so on, in each direction. The fixed readers, thus, form a grid-like structure over the area to be covered. Each cross section of the grid forms the rectangular region within which one, or more, mobile readers move as per the *zig-zag* mobility model. A more detailed pseudocode is given in Figure 4.3

4.5 Optimizing Objective Function

The layout generating heuristic produces a set of layouts. For each layout, the number of fixed and mobile readers, the performance metrics like the *Cost* and *TRT*, are calculated

```
1: Define: $l$ = length of the aisle, $d$ = inter aisle distance, $X, Y$ = dimensions of the area
   to be covered, NMR = number of mobile readers, NFR = number of fixed readers, R =
   interrogation range of fixed readers.
2: Assumption: The length of the aisle is along $X$.
3: Initially: NMR = NFR = 0 and MRInitialPosition, FRPosition are both empty arrays.
4: for ($d1 = l + d$; $d1 \leq X$; $d1 = d1 + l + d$) do
5:   for ($d2 = d$; $d2 \leq Y$; $d2 = d2 + d$) do
6:     for ($i = 0$; $i \leq X$; $i = i + d1$) do
7:       MRInitialPosition = FRPosition = new empty lists.
8:       for ($j = 0; j \leq Y; j = j + d2$) do
9:         FRPosition[NFR++] = $(i, j)$//forms a column of readers $d2$ apart
10:      end for
11:      if ($j \neq (Y + d2)$) then
12:        FRPosition[NFR++] = $(i, Y)$
13:      end if
14:    end for
15:    if ($i \neq (X + d1)$) then
16:      for ($j = 0; j \leq Y; j = j + d2$) do
17:        FRPosition[NFR++] = $(i, j)$
18:      end for
19:      if ($j \neq (Y + d2)$) then
20:        FRPosition[NFR++] = $(i, Y)$
21:      end if
22:    end if
23:    for ($i = 0$; $i < X$; $i = i + d1$) do
24:      for ($j = 0; j \leq Y; j = j + d2$) do
25:        MRInitialPosition[NMR++] = $(i + R, j)$
26:      end for
27:    end for
28:    This is one layout. It uses NMR mobile readers and NFR fixed readers and their
       positions are given by arrays FRPosition and MRInitialPosition.
29:  end for
30: end for
```

Figure 4.3: LGH_1 heuristic

and those conforming to the input constraints are retained. One of these is chosen for recommending to the user, as per an appropriate optimizing objective function.

In our supermarket application, we see that both these metrics - *Cost* and *TRT* - are important. We would like to use minimum number of readers and yet provide complete coverage as often as possible. So, the layout which is minimal in both *TRT* and *Cost* needs to be chosen. Hence, we use the *least square sum* as the objective function and apply it on the parameters *TRT* and *Cost*. Thus, the layout with minimum value for $TRT^2 + Cost^2$ is chosen as the optimal one.

For some applications, the *TRT* may be a more important than the *Cost*, while for others the *Cost* may be more important than *TRT*. In such cases, other objective functions could be used to select the optimal layout.

4.6 Screenshots

Some sample screenshots to show the operation of the *RFIDcover* has been included in Figures 4.4 to 4.6.

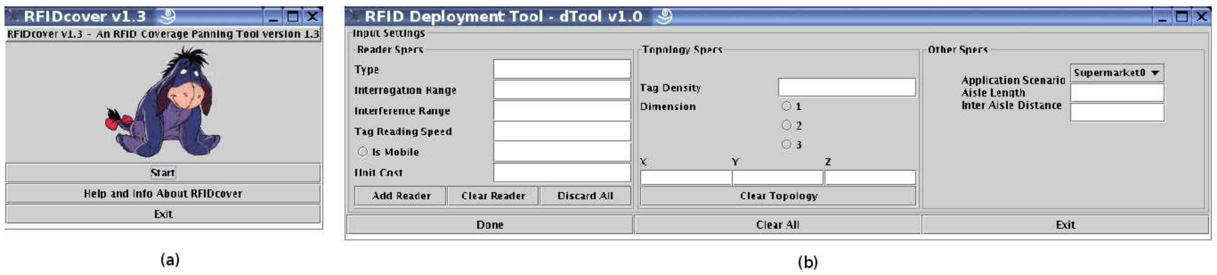
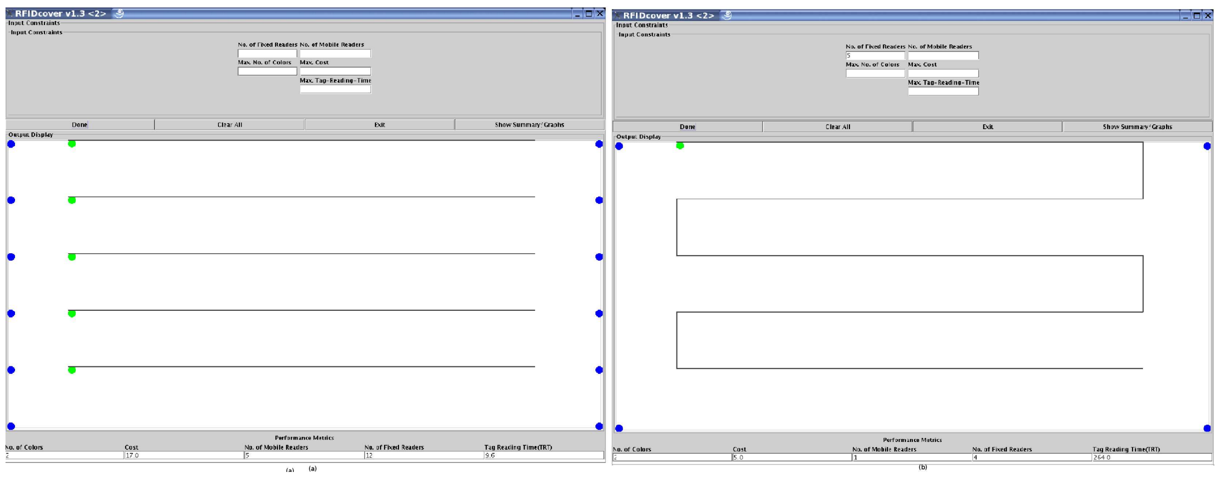


Figure 4.4: Screenshots of Input to *RFIDcover*

Figure 4.4 (a) shows the start-up screen. The input specifications are provided by the user as shown in Figure 4.4 (b).

Figures 4.5 and 4.6 shows the output provided to the user. The details of the best layout is shown in screenshot of Figure 4.5. The green (or light gray) circles represent mobile readers and the blue (dark gray) ones represent the fixed ones. The line gives the path followed by the mobile reader. The best layout chosen, for an example run (Table 5.1), is displayed in Figure 4.5 (a). Another layout that is shown in Figure 4.5 (b) is the layout with the worst *TRT*, where a single mobile readers moves in a zig-zag fashion to cover the given area. This was obtained by putting an additional constraint on number of readers.

The output graphs that summarize the layouts for the user is as shown in Figure 4.6.



(a) Performance Metrics

No. of Colors	Cost	No. of Mobile Readers	No. of Fixed Readers	Tag Reading Time(TRT)
2	17.0	5	12	9.6

(b) Performance Metrics

No. of Colors	Cost	No. of Mobile Readers	No. of Fixed Readers	Tag Reading Time(TRT)
2	5.0	1	4	264.0

Figure 4.5: Screenshots of Output of *RFIDcover*

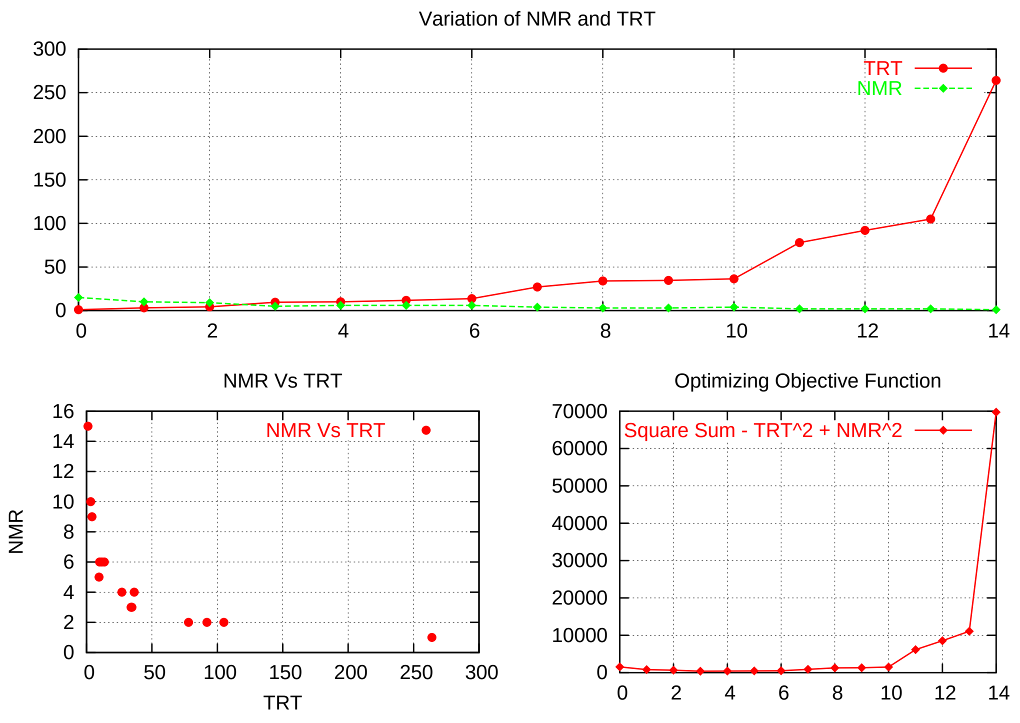


Figure 4.6: Screenshots of Output of *RFIDcover*

The first graph shows how *TRT* and the *number of mobile readers (NMR)* vary among the layouts conforming to the constraints. The second graph plots the *TRT* Vs *NMR* graph. Each point in it represents a conforming layout. Looking at it we can see that the layout nearest to the origin would be a better choice for the supermarket scenario than those points near to the axes. Thus, the *least square sum* optimizing objective function is appropriate. The third graph is the result of applying the optimizing objective function to each layout. The best one is picked from it.

In the next Chapter, we discuss the evaluation of the implementation for the supermarket application scenario.

Chapter 5

RFIDcover Evaluation

In this section, we discuss the correctness and usability of *RFIDcover*'s current implementation for the supermarket inventory application scenario. At first, we show how using mobile readers results in a cost-effective coverage of an area. Then, we evaluate how closely the *zig-zag* mobility model follows the *sufficient bound* (Section 2.2) on the number of mobile readers required to provide complete coverage of an area.

5.1 The Example

We consider the example shown in Table 5.1. We ran *RFIDcover* on this example inputs, to get the data used to plot the graphs in Figures 5.1 and 5.2 that are discussed in the rest of the sections.

Reader Spec		Topology Spec		Application	
Interrogation Range:	$2m$	Dimension X:	$24m$	Supermarket Scenario	
Interrogation Range:	$2.5m$	Dimension Y:	$15m$	Aisle Length:	$5m$
Tag Reading Speed:	$70 tags/s$	Tag Distribution:	Uniform	Inter Aisle Distance:	$3m$
Unit Cost:	1	Tag Density:	$5/m^2$	MAC mechanism	
Max. Speed:	$5m/s$	Aisle length along X		Static Coloring	

Table 5.1: The example

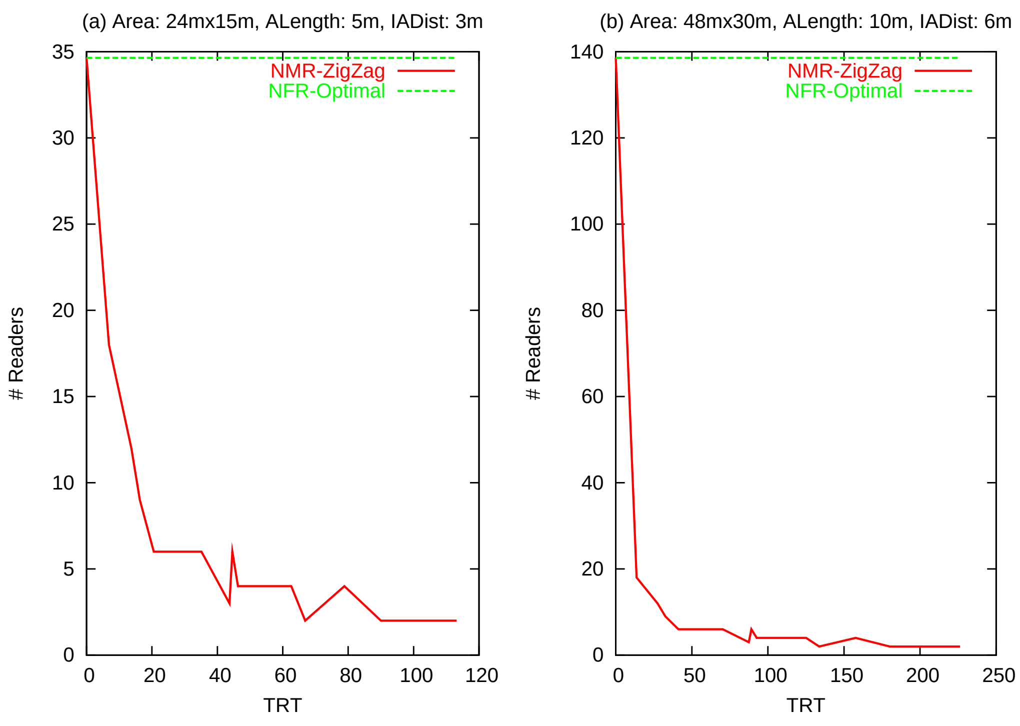


Figure 5.1: NMR for Zig-zag Mobility Model Vs NFR

5.2 Fixed Vs Mobile Readers

Figure 5.1 shows the graphs of the number of mobile readers (NMR) and the optimal number of fixed readers (NFR) with varying *TRT*. From the graphs in the figure, we can see that for providing complete coverage of an area, the number of mobile readers needed is much lesser than the number of fixed readers. The number of mobile readers required decreases drastically as *TRT* increases and would reach 1 when *TRT* approaches ∞. This is as expected.

For practical applications, very high values of *TRT* may not be meaningful. Hence, the region of interest is the area of the graph near the origin, where the *TRT* values are within the range of a few seconds. Here also, we find that except when the *TRT* is negligibly small, the number of mobile readers required drops significantly with slight increase in *TRT*. Hence, using mobile readers for such values of *TRT* would be cost-effective.

Another important observation is regarding the slope of the curve for the number of mobile readers in this region. The slope of the curve is very steep if $r \ll X$, or Y. On

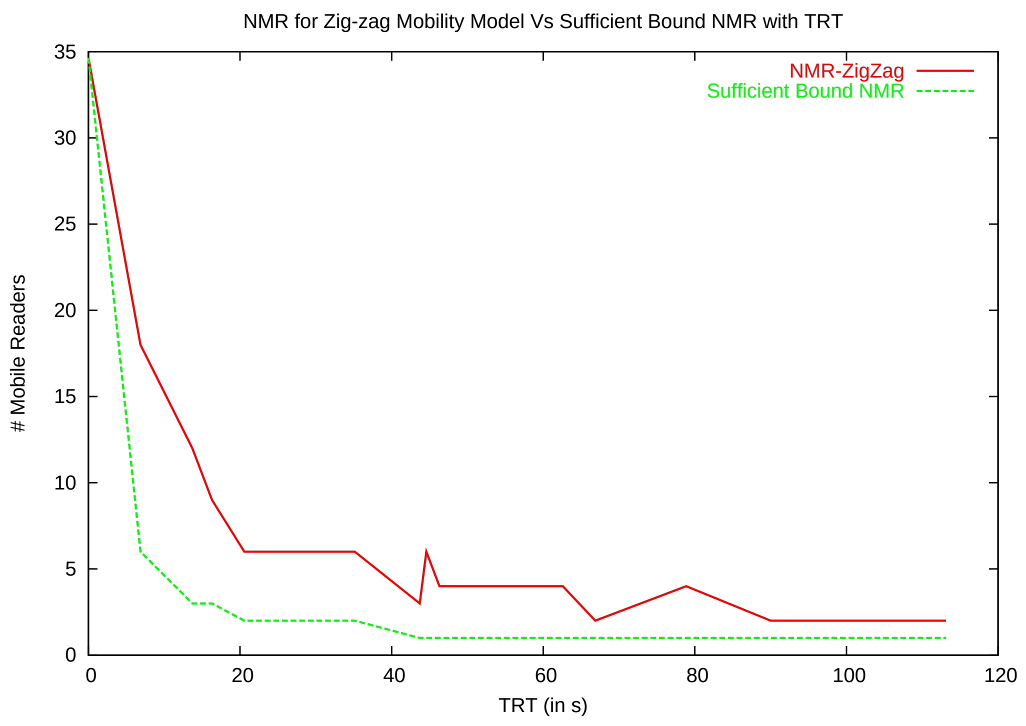


Figure 5.2: NMR for Zig-zag Mobility Model Vs Sufficient Bound NMR

the other hand, if r is comparable to X and Y, the slope of the curve falls at a much lower rate. For example, the curve in Figure 5.1 (a) (which corresponds to an area of dimensions $24m \times 15m$ and *aisle length* $= 5m$ and *inter aisle distance* $= 3m$), falls from 35 readers to around 11 readers as the *TRT* increases from $0s$ to $15s$. On the other hand, the curve in Figure 5.1 (b) (which corresponds to an area of dimensions $48m \times 30m$ and *aisle length* $= 10m$ and *inter aisle distance* $= 6m$), falls from 139 readers to 17 readers for the same increase in *TRT* ($0s$ to $15s$), even though the number of aisles is the same in both cases.

5.3 Zig-zag Mobility Model Vs Sufficient Bound

The graph in Figure 5.2 compares how the number of mobile readers (NMR) needed for complete coverage using the *zig-zag* mobility model with respect to the *sufficient bound* on the number of mobile readers. As can be observed, the *zig-zag* mobility model curve follows the *sufficient bound* quite closely. This asserts the practical suitability of the zig-zag mobility model for the supermarket application scenario.

In the next Chapter, we would look at some possible extensions of *RFIDcover*.

Chapter 6

RFIDcover Extentions

The architecture of *RFIDcover* as discussed in Chapter 3 is generic and extendible. This makes it possible to implement other application scenarios. Also, for existing application scenarios, other mobility models, layout generating heuristics and MAC mechanism can be implemented. In this Chapter, we discuss how *RFIDcover* can be extended to support a slight variant of the supermarket application scenario. The implementation, and/or evaluation, of these suggested models was beyond the scope of this project.

6.1 Retail Inventory Tracking Application Variant

Consider the retail inventory tracking application introduced in Chapter 1 and used as an illustrative example throughout this report. Now, assume that the supermarket has aisles along both length and breadth of the area. This is the application under consideration.

6.1.1 Mobility Model and Layout Generating Heuristic

For this application the *zig-zag* mobility model will not be suitable. In this scenario, the *to-and-fro* mobility model, mentioned in Section 3.3, will be more appropriate. Using the *to-and-fro* mobility model, the mobile readers will cover the aisles by moving in a straight line along aisles in both dimensions of the area.

Now, LGH_1 described in Section 4.4 will not be suitable for the *to-and-fro* mobility model as it does not generate a layout that covers the aisles both along length as well as breadth of the area. So, another heuristic, say LGH_2, is needed. This heuristic would be similar to LGH_1 except that it would place mobile readers along both dimensions of the area to be covered.

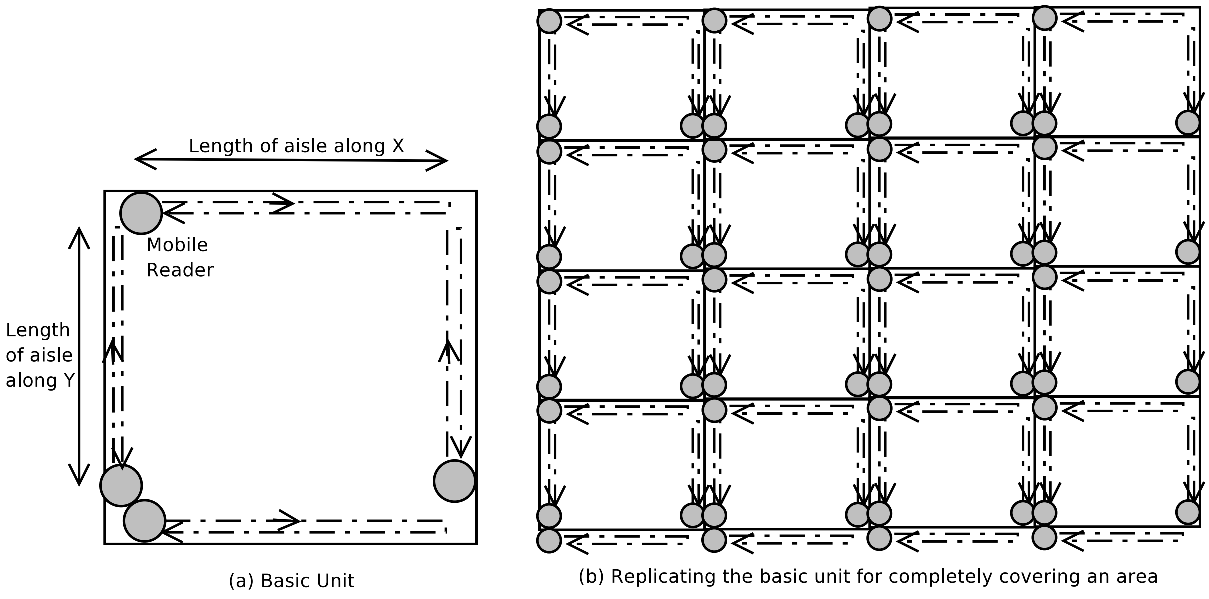


Figure 6.1: The To-and-Fro Mobility Model

Figure 6.1 shows a sample layout generated using LGH_2, and the readers moving with the *to-and-fro* mobility model completely covering a given area.

6.1.2 MAC Mechanism

This application can use the *static coloring MAC* itself. Assuming the *aisle length* is greater than the *interference range*, if the mobile readers move using the *to-and-fro* mobility model and the layouts are generated by LGH_2, this requires three time slots for it's operation - one for fixed readers and two for the mobile readers.

So, we see that in order to implement a new application all we need is to first implement at least one mobility model, a layout generating heuristic and a MAC mechanism that supports the new application. Thus, *RFIDcover* is a flexible tool which can be extended easily to support many other applications.

6.2 Dynamic Coloring MAC

Although *static coloring MAC* has the advantage of being simple and easy to implement, it assigns slots based on worst case scenario. Thus, it may not always be efficient if the readers are hardly ever in the worst case scenario.

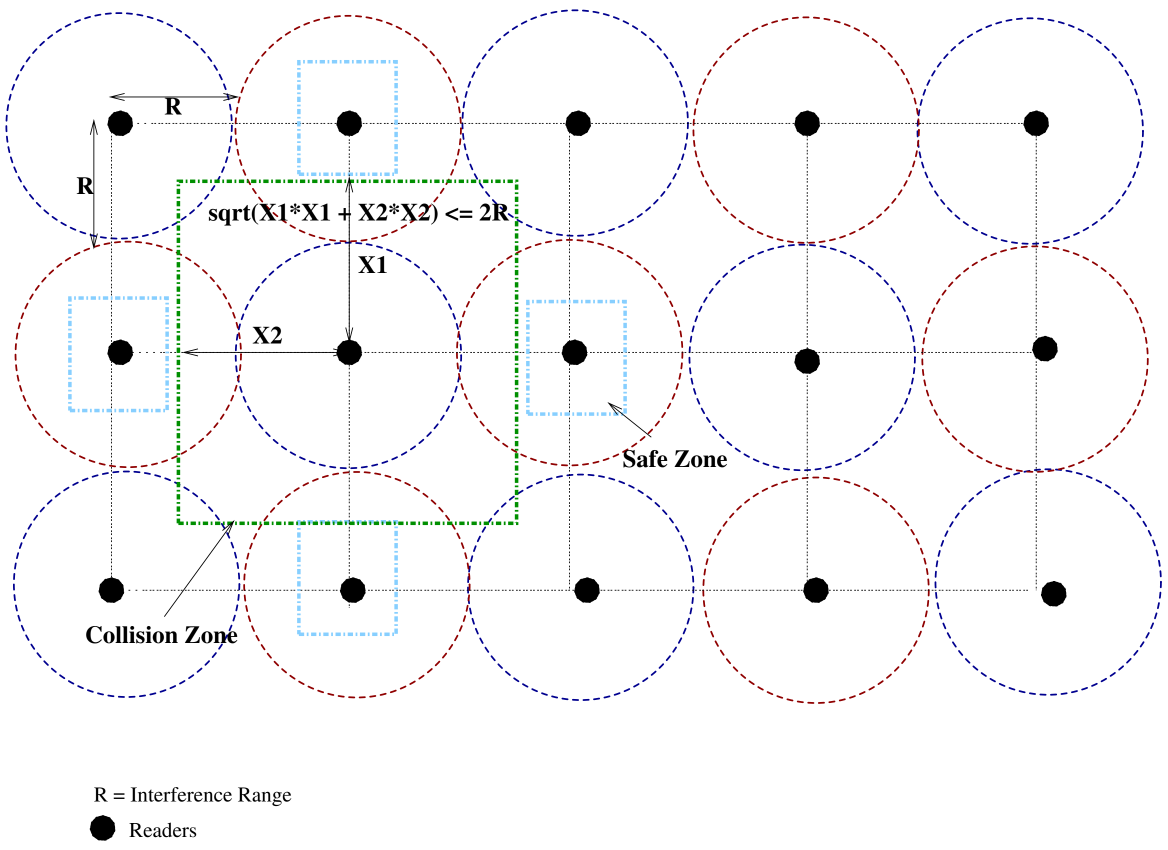


Figure 6.2: Dynamic Coloring MAC - The Collision Zone

An alternative is to initially start with some assignment of slots and adapt to the changes as and when they occur. We propose the *dynamic coloring MAC* which uses this approach. It starts with minimum number of slots assigned to the readers initially. At this point, the reader network is said to be in `Min-Color-Mode`. As the readers move, they come into the interference range of each other, we call it as the `Collision Zone` (See Figure 6.2). In this case, the readers switch to a higher number of slots. The network then enters the what we call the `General-Color-Mode`.

Such a scheme would work in scenarios where most often than not, the network would be in `Min-Color-Mode`, entering the `General-Color-Mode` only occasionally. Even in the case of the `General-Color-Mode` the number of slots taking a very high value is highly unlikely. Also, the number of slots required for operation would depend on the probability of the network being in each `Color-Mode`. That is,

$$noSlots = 1 * P_1 + 2 * P_2 + 3 * P_3 + ... + m * P_m$$

where $m = total\ number\ of\ readers$ and P_i is the probability that the system is operating with i colors.

Thus, if the nodes are highly mobile, the probability of collisions increases, thereby making it reasonable to have worst case assignment of slots. Whereas, when the mobility of the nodes is very less, there would be occasional switching, and the dynamic coloring scheme could improve the performance in terms of time efficiency.

However, the major difficulty in such a scheme is to know when to switch between the `Min-Color-Mode` and the `General-Color-Mode`, and how? In the VDCS mode mentioned in Colorwave[7], on experiencing collision each reader in the system picks a color randomly. In dynamic coloring scheme we attempt to eliminate that randomness. However, considerable overhead will be incurred to enable such switching of mode.

A natural next step would be to implement the dynamic coloring scheme, and integrate it with *RFIDcover*. Once integrated, the scheme could be compared to the static coloring scheme for a given topology configuration.

6.3 Covering 3D Space

Consider an application where the tagged items are present in a three dimensional space. An example could be a large warehouse where tagged items are placed on shelves that are high enough to reach the ceiling. In such scenarios, we need to cover a 3D space instead of the 2D area discussed so far. Implementing layout generating heuristics and mobility models to support such an application could be another extension of *RFIDcover*.

In order to cover a 3D space, as was done in a 2D area, we would consider the dimensions of the minimum-volume cuboid enclosing the region to be covered as the input. We could then cover this region as follows. A variant of the LGH_1 and LGH_2 layout generating heuristics could be used to generate a layout with the readers placed in each of the three dimensions of the region. The mobile readers could follow the *to-and-fro* mobility model and cover the entire region by moving in each dimension.

6.4 Other Limitations

Currently we make the following assumptions:

- We do not consider any environmental effects and assume that the readers have a circular range.
- We consider a homogeneous system, wherein all the fixed/mobile readers have the same characteristics. Variable power devices are available that can be used to cover an area, making the system heterogeneous.

These assumptions could be a limit to *RFIDcover*'s functionality. Overcoming these limitations is another way to extend the present implementation of *RFIDcover*.

Chapter 7

Conclusions

We have seen that for many practical RFID applications, where complete coverage is required only on a periodic basis, using mobile readers instead of fixed readers is a cost-effective option. This is specially true for applications such as retail inventory tracking.

We presented *RFIDcover*, an automated coverage planning tool that determines an optimal layout of readers required to guarantee complete coverage for a given application scenario. We have also evaluated the effectiveness of *RFIDcover* for the retail inventory tracking application. *RFIDcover* provides the user with crucial deployment-specific information such as the number of readers needed, their placement, movement pattern, and the deployment cost. It also gives the user the flexibility to input additional constraints on-the-fly, thereby making it a very useful tool for RFID deployment.

The architecture of *RFIDcover* is generic and extendible, enabling easy implementation of other application scenarios. Even for existing application scenarios, other mobility models, MAC mechanisms and layout generating heuristics can be implemented. This would enable a comparison of various deployment options for the same application.

Bibliography

[1] Klaus Finkenzeller. *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification.* Chichester : John Wiley, Leipzig, dritte edition, 2003.

[2] Radio Frequency Identification - A Basic Primer. White Paper, AIM Inc WP-98/002R2, August 2001. `http://www.aimglobal.org/`

[3] `http://mathworld.wolfram.com/CirclePacking.html`.

[4] Richard Kershner. The number of circles covering a set. In *American Journal of Mathematics*, volume 61, page 665, July 1939.

[5] Yi Guo and Zhihua Qu. Coverage Control for a Mobile Robot Patrolling a Dynamic and Uncertain Environment. *Proceedings of World Congress on Intelligent Control and Automation*, June 2004.

[6] Daniel W. Engels. The Reader Collision Problem. Technical report, `http://www.epcglobal.org`, 2002.

[7] J. Waldrop, D. W. Engels, and S. E. Sarma. Colorwave: An anticollison algorithm for the reader collision problem. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.

[8] Draft paper on characteristics of rfid-systems. White Paper, AIM Inc WP-98/002R2, July 2000.

[9] A basic introduction to rfid technology and its use in supply chain. Technical report, Laran Technologies, January 2004.

Acknowledgements

I express my sincere gratitude towards my guide **Prof. Sridhar Iyer** for his constant support and encouragement. His invaluable guidance has been instrumental in the successful completion of the project work.

I would like to thank members of the **Mobile Computing Research Group** at KReSIT, and also my batchmates B. Nagaprabhanjan, Charu Tiwari and Shailesh M. Birari for their valuable suggestions and helpful discussions.

Last but not the least, I would like to thank the entire KReSIT family for making my stay at IIT Bombay a memorable one.

S. Anusha
I. I. T. Bombay
June 30^{th}, 2005