RFIDPlanner - A Coverage Planning Tool for RFID Networks

Dissertation

submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Technology and Master of Technology

by

Nitesh Gupta

(Roll no. 01D05011)

under the guidance of

Prof. Sridhar Iyer

Computer Science and Engineering Department

Indian Institute of Technology Bombay

2006

Dedicated to my family

Dissertation Approval Sheet

This is to certify that the dissertation entitled

RFIDPlanner **- A Coverage Planning Tool for RFID Networks**

by

Nitesh Gupta

(Roll no. 01D05011)

is approved for the degrees of

Bachelor of Technology and Master of Technology

____

Prof. Sridhar Iyer

(Supervisor)

____

Prof. Anirudha Sahoo

(Internal Examiner)

____

Mr. Swapnil Paranjpe

(External Examiner)

____

Prof. Abhay Karandikar

(Chairperson)

Date: ______________

Place: ______________

Abstract

Radio Frequency Identification or RFID systems are emerging as economical solutions for fast identification of objects. One of the important aspects of setting up a RFID network is deploying RFID readers to ensure complete coverage of RFID tags in a given area. This task is usually accomplished by conducting site surveys and then deploying RFID readers in different configurations to determine the optimal one. As the size of the given area increases, time taken to setup a RFID network increases thus increasing the cost of deployment.

We present *RFIDPlanner*, a coverage planning tool for RFID networks, which attempts to tackle the issue using simulation. Given the details of an area, the properties of obstacles present in the area, and the specifications of the RFID reader and tags used, *RFIDPlanner* simulates the coverage pattern of a reader giving a graphical view of the RFID reader's interrogation range. Through its interactive GUI the layout can be modified during run-time, by moving and deleting existing RFID readers and adding new ones. The zoom feature allows different levels of visualizations ranging from a view of the entire layout to focused views of particular sections and RFID readers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Radio Frequency Identification, or RFID, is a generic term used for technologies that use radio waves to automatically identify individual items. RFID Technology is the next generation identification technology set to succeed barcodes. Unlike barcodes this technology does not require line of sight for object identification and can identify objects over a range using radio waves. The operations involved are performed using low cost components. Thus, this technology makes identification process much faster, capable of identifying hundreds of products in few seconds, making it ideal for businesses requiring identification of huge number of objects in a short time [1].

A Radio Frequency Identification (RFID) System consists of RFID tags and RFID readers at its core, performing the main task of identifying individual objects using radio waves. An overview of RFID tags and RFID readers along with the applications is presented in the following sections. This is followed by the definition of the reader coverage planning problem that we attempt to address, related work in coverage planning in indoor WLANs and a proposed solution outline.

1.1 RFID Tags

An RFID tag is a microchip coupled with a small antenna and is tagged to the object to be identified. The microchip carries the data associated with the object it is tagged to, usually a serial number also known as the Tag ID. The tag responds to the reader interrogation by modulating its data on the interrogation signal and using it for communication with the reader. A few tags are shown in Fig. 1.1. RFID tags can be classified on the basis of power supply and memory.

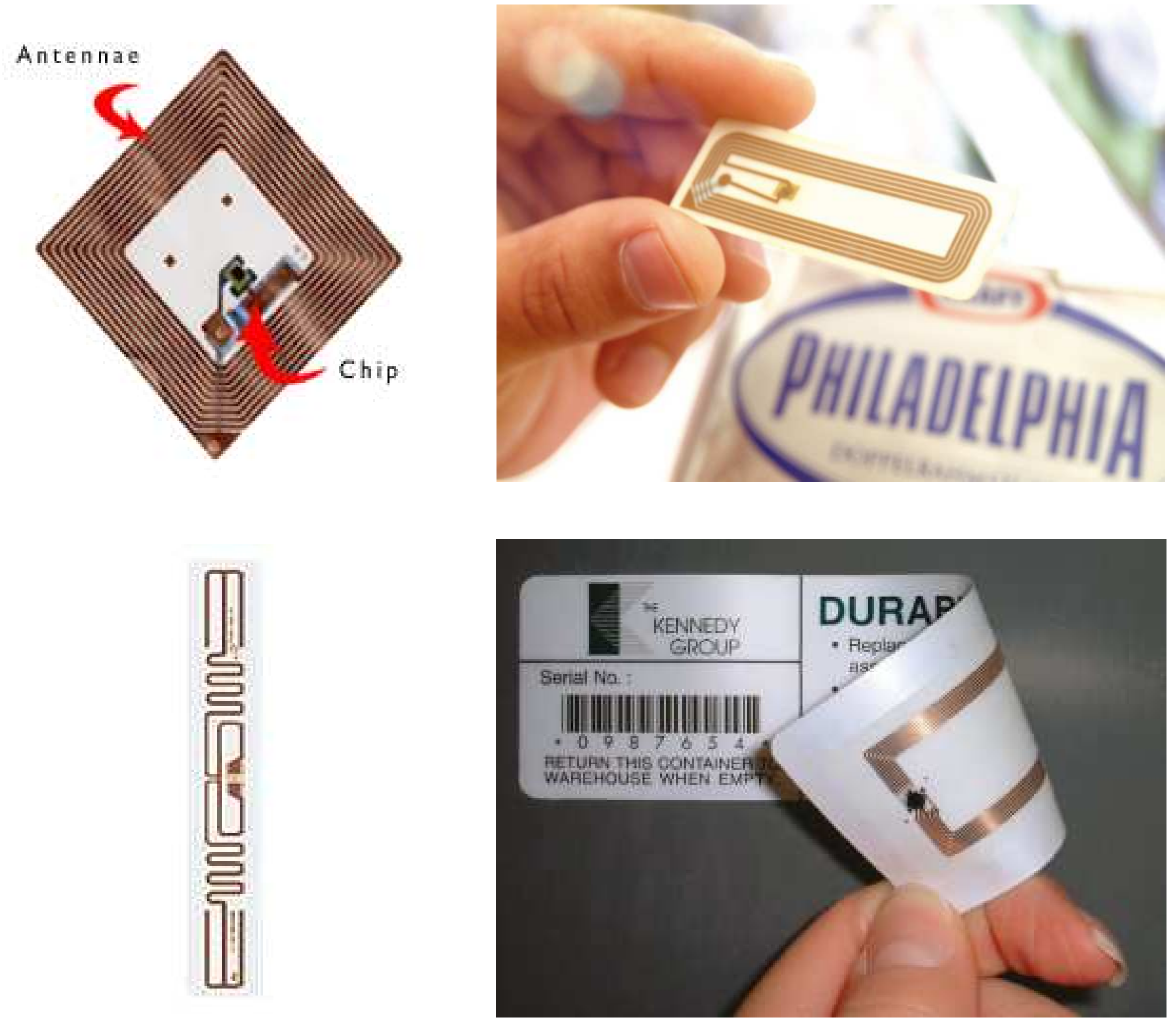


Figure 1.1: RFID Tags

1.1.1 Power Supply

An important factor in determining the applicability of RFID technology is the source of power for a tag. Tags can be either passive or active.

1. **Passive Tags**: Passive tags derive power from the interrogation signals broadcasted by the reader. These tags contain a separate unit (containing the antenna) on the microchip which is responsible for deriving the power as well as transmitting data. The dependence on the interrogation signal for the power to operate leaves the tags with a much smaller range of communication as compared to active tags, ranging from 4 inches to 15 feet [1]. These tags have low storage capacities, ranging from a few bits upto 1 KB and are usually write-once-read-only or read-only tags. Passive tags, because of the comparatively lower cost, are suitable for retail businesses involving large number of objects to be identified.

2. **Active Tags**: Active tags are equipped with onboard power supply and hence have a longer transmission range of upto 300 feet [1]. They have higher storage capacities of upto 512 KB and typically can be re-written by RFID readers. Power supply increases the cost of the tags, limiting their applicability. For e.g. active tags might be used for tracking and theft prevention of expensive equipments.

1.1.2 Memory

The memory space available on the tag can be varied according to requirements. Tags used for identification for large number of items would usually have the minimum possible memory required since as the memory space increase the cost of the tag increases. Tags can be classified on the basis of read and write operations available on the tag memory.

1. **Read Only**: The memory is one time programmable (OTP). The tag information, which is usually the tag ID, is stored in the tag during the time of manufacturing. No further write operations are allowed.

2. **Write Once Read Only**: The user is allowed to enter information in the tag memory only once. After this no further writes are allowed and tags can only be read.

3. **Read Write**: Any number of read write operations on the tag memory is allowed. Such memory is usually an EEPROM.

1.2 RFID Readers

An RFID reader consists of a radio frequency module for two way communication with tags along with a control unit. It interrogates the tags or reads the information stored in the tags by transmitting radio frequencies which are reflected back by the tags along with the data.

Readers are usually connected through middleware to a back-end database which stores processed information passed by the middleware and might also contain detailed information about the objects being identified. Readers can typically read 100 - 300 tags per second [1].

Readers can be installed at fixed locations such as entry and exit points in ware houses, or at point of sales in shops. Readers can also be mobile in form of handheld devices. Fig. 1.2 shows a few RFID readers.

1.3 Applications

RFID technology promises to make a big impact on data intensive business processes such as supply chain, stock management and warehousing which require rapid identification of a huge number of objects. RFID technology provides a fast mechanism for object identification, increasing efficiency with lower error rate. Other applications of this technology include equipment and livestock tracking as well as security purposes, such as tracking of and theft prevention in automobiles. Table 1.1 lists some of the business areas where RFID systems have potential applications [1].

1.4 Problem Definition

Since RFID readers have small interrogation ranges, deploying an RFID network involves large number of RFID readers. To be cost-effective, it becomes important to deploy minimum number of readers while ensuring complete coverage of the area where RFID tags

Figure 1.2: RFID Readers

Table 1.1: RFID Applications (Source: Sridhar Iyer [1])

Business Area	**Applications**
Manufacturing and Processing	- Inventory and production process monitoring - Warehouse order fulfillment
Supply Chain Management	- Inventory tracking systems - Logistics Management
Retail	- Inventory control and customer insight - Auto checkout with reverse logistics
Security	- Access control - Counterfeiting and theft control/prevention
Location Tracking	- Traffic movement control and parking management - Wildlife/Livestock monitoring and tracking

might be present. The procedure to determine such an optimum configuration typically involves

- manual site surveys by surveyors who have extensive knowledge of radio wave characteristics and behaviors patterns in presence of different materials,
- testing of different reader positions and tracking reader range patterns, which change with a slight change in reader positions or a new obstacle in the reader's range,
- significant time and monetary costs related with manual site surveys.

Thus, deploying RFID readers in a fast and cost effective way is a difficult problem and requires automated tools designed specifically for this purpose.

1.5 Related Work: Coverage Planning In Indoor WLANs

The problem of coverage planning in RFID networks can be thought of as similar to coverage planning in indoor WLANs. Just like RFID readers in RFID networks, access points in WLANs need to be placed optimally to ensure complete coverage of a given area taking into account the different obstacles present in the area. Several simulation tools have been developed to aid coverage planning using propagation modeling. Some examples of such tools are WISE developed at AT&T [7] and SpectraGuard Planner developed by AirTight Networks [8]. A brief explanation of both the tools is given below.

1.5.1 WISE

Wireless Systems Engineering (WISE) is a tool developed at AT&T Bell Laboratories for designing indoor and campus-sized wireless systems [7]. It uses 3-D ray tracing techniques to model radio waves and uses computational geometry and optimization algorithms to optimally place base-station transceivers. It provides three types of functionalities to aid in coverage planning.

(i) *Prediction*: Given the building details (wall locations and composition) and user-specified base-station locations, WISE can determine the received signal power through out the building using 3-D ray tracing techniques. This helps in evaluating performance for a specified set of base locations and parameters. CAD is used for the purpose of modeling buildings and then extracting appropriate physical data required by WISE for the predictions.

(ii) *Optimization*: Given initial set of parameters and user-specified base locations (which can also be randomly selected), WISE uses prediction to compute a locally optimum solution. This is done by moving the base stations according to a modified form of Nelder-Mead direct search algorithm to improve the predicted search so as to maximize the building area where signal-strength requirements are met.

(iii) *Interaction*: The WISE user interface displays plan, elevation and perspective views of a building, and shows signal strength through out the building. Various parameters can be adjusted, and the locations of base stations and wireless mobile devices can be set interactively.

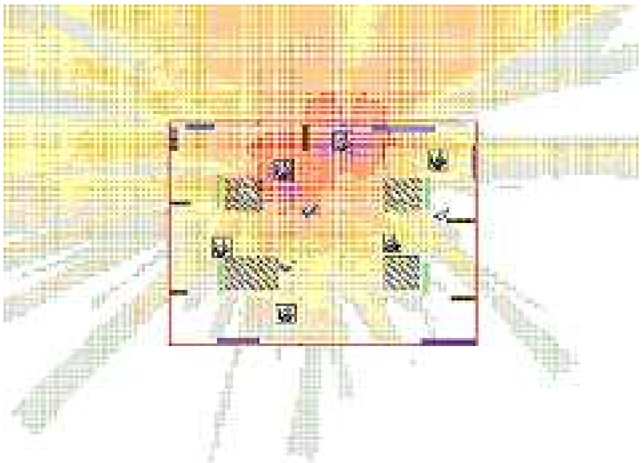
WISE is a fast tool, based on deterministic modeling of radio waves, to evaluate and optimize performance of indoor WLANs, with relatively low rates of error.

1.5.2 SpectraGuard Planner

SpectraGuard Planner is a WLAN planning coverage tool developed by AirTight Networks [8]. It has been developed using several principles similar to those used for WISE. For example, like WISE, it follows a deterministic modeling technique of 3-D ray tracing to predict the received signal strength in different parts of the given building. The physical data of the buildings under consideration is entered using CAD. SpectraGuard Planner also provides a user interface which offers a 2-D visualization of base-station coverage, unlike WISE which offers a 3-D visualization.

SpectraGuard Planner, however, does not provide any optimization mechanism as is available in WISE. Thus, it is upto the user to try different placements of base-stations to come up with an optimal configuration. On the other hand, SpectraGuard Planner provides security coverage and risk level assessment for a given setup of WLANs. Few snapshots of SpectraGuard Planner generated coverage patterns are shown in Fig. 1.3.

(a) Access Point Coverage Overview



(b) Access Point Spillage View

Figure 1.3: SpectraGuard Planner Snapshots

1.5.3 Indoor WLANs vs. RFID Systems

It is not easy to adapt WLAN coverage tools for RFID networks since there remain some basic differences between the two. To remain cost-effective RFID networks are constrained to use passive RFID tags that depend upon the RFID reader for their power supply. Hence there is a limit on the interrogation range of a RFID reader. On the other hand an access point interacts with wireless devices such as laptops, PDA, etc. which have superior processing capabilities, higher tolerance to communication errors and a power supply of their own. These devices are thus capable of communicating with access points over relatively larger distances.

These differences make it difficult to use tools designed for WLANs for coverage planing in RFID networks. Therefore, there is a need to build specific coverage tools for the RFID domain. To the best of our knowledge there is only one tool, *RFIDCover*, that has been specifically designed for RFID systems. This is discussed in detail in the following section.

1.5.4 *RFIDCover*

RFIDcover is an automated coverage planning tool that addresses the problem of providing periodic complete coverage of a given area [6]. For example, applications such as retail inventory tracking need to track tags periodically, say each tag being covered every τ seconds. *RFIDCover* attempts to use mobile RFID readers to cover the area so as to be more cost effective.

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.

Although *RFIDCover* provides coverage solutions using a mix of fixed and mobile readers, there remain certain constraints in the tool which limit its applicability. *RFIDCover*

(i) does not take into consideration the effect of different obstacles on a RFID reader's range. It assumes that the range is perfectly circular even in presence of obstacles, such as walls, doors, etc., which might not be the case in every scenario.

(ii) provides complete coverage only periodically and not continuously, and hence cannot be used in applications in which tags need to be monitored continuously. It also relies on mobile RFID readers for complete coverage, an option which might not be available always.

(iii) provides complete coverage of the given area, regardless of the fact that some portion of that area might not contain any tags, thus leading to wastage of resources. It does not give the flexibility of choosing between covering the entire area and covering only those portions of the area where tags are present.

Given the above limitations of *RFIDCover* we decided to design a new tool, *RFIDPlanner* which would provide a more flexible approach for coverage planning and deliver more accurate results.

1.6 Solution Outline

To address the problem of coverage planning we have designed and developed *RFIDPlanner* - a coverage planning tool. Given the details of the area to be covered, properties of the obstacles present in the area, and reader and tag specifications, *RFIDPlanner*

(i) emulates a reader's range attenuation patterns,

(ii) provides an interactive layout visualization and modification unit to visualize range attenuation patterns in different locations and modify reader layout configurations during runtime,

(iii) eliminates the need of site surveys while speeding up testing of different configurations, thus bringing down both cost and time of deployment.

The *RFIDPlanner* architecture has been designed and developed in a flexible manner to allow changing configuration of different modules as well as addition of independent modules to enhance the functionality of the tool. For example, the architecture allows addition of independent modules which can generate sample configurations of reader placement according to given scenarios. These configurations can then be visualized using *RFIDPlanner* and appropriately modified to suit requirements.

1.7 Thesis Organization

In Chapter 2 we discuss the basics of *RFIDPlanner*: assumptions, design drivers, propagation modeling used and range calculation formulae. The architecture and implementation of *RFIDPlanner* has been discussed in detail in Chapter 3 wherein the different modules and range emulation algorithm have been explained, which is followed by a discussion on implementation of *, RFIDPlanner*.

Chapter 4 talks about two different heuristics, based on a case study of shopping mall scenario, that have been designed to work with *RFIDPlanner* to generate reader layout configurations.

Chapter 5 discusses PINES^{TM}, an RFID middleware, and how *RFIDPlanner* can be used in combination with PINES^{TM} to provide a one-stop solution platform for deploying and maintaining RFID systems.

Chapter 2

RFIDPlanner Design Basics

In this chapter we describe the assumptions, design drivers, propagation modeling and antenna range calculations involved in the design of *RFIDPlanner*.

2.1 Assumptions

Some of the assumptions made while designing *RFIDPlanner* are stated below.

- RFID readers being used are of the same type, having same characteristics and hence the same maximum interrogation range.
- RFID tags being used are passive and of the same type.
- Antennas are isotropic having circular ranges.
- Obstacles are rectangular and are either aligned along the x-axis or along the y-axis.

2.2 Design Drivers

RFIDPlanner has been designed keeping in mind the following objectives.

- Place readers statically to cover the given area
- Simulate the interrogation range attenuation patterns for RFID readers in presence of different obstacles.
- Provide a visualization tool that displays the layout of readers in the given area.
- Allow users to modify the layout during run-time allowing them to move and delete existing RFID readers and add new ones.

Table 2.1: Attenuation values

Type of Obstacle	Attenuation Value (dB)
Wooden partition	1
Brick wall	6

2.3 Propagation Modeling

Propagation models can be classified as deterministic and empirical [3]. Deterministic modeling is based on electromagnetic wave propagation theory and makes use of ray tracing techniques to model range patterns. It requires detailed description of the area under consideration which can often be complicated and difficult to obtain.

On the other hand, Empirical modeling makes use of statistically processed representative measurements of propagation losses due to different obstacles that lie in the path of the electromagnetic waves. Empirical modeling assumes that effects of reflection, diffraction and scattering on the range patterns are taken care of in the representative measurements. The propagation loss between a transmitter and a receiver L [3] is given by the following equation

$$L = L_{FSL}(d) + \sum_{i=1}^{N} k_{wi} L_{wi} \tag{2.1}$$

where L_{FSL} (dB) is the free space loss for distance d(m) between transmitter and receiver antennas, k_{wi} is the number of walls of i-th type between transmitter and receiver antennas, L_{wi} is attenuation factor for i-th wall type and N is the number of wall types. Since we would be looking at a 2D model of the area, floor attenuations are not taken into consideration.

As RFID readers have small interrogation ranges, the effects of reflection, diffraction and scattering would not be substantial enough over the small area of coverage of a single reader. It would suffice to use representative measurements of propagation losses for calculating the range patterns. Therefore, empirical approach has been taken for propagation modeling in *RFIDPlanner* instead of a deterministic approach. We use a table similar to one shown in Table 2.1 for propagation loss calculations. More attenuation values for different type of materials can be found in [4] and [5].

2.4 Antenna Range

The interrogation range for an antenna, in absence of any obstacles, is calculated on the basis of power supplied by its parent reader. The power received at a distance r from the source, i.e. the electromagnetic coupled antenna, is inversely proportional to the square of that distance r^2 [2]. For a given power the interrogation range of an antenna is defined as the maximum distance r at which the received power is just sufficient to make the tag operational, i.e., the tag is able to process the RF signal and reflect back information.

To calculate the power received at a distance r we use the formula for free space path loss a_F, which is a measure of the relationship of power emitted by an antenna into free space and the power received at the tag [2].

$$a_F = -147.6 + 20log(r) + 20log(F) - 10log(G_T) - 10log(G_R) \tag{2.2}$$

where f is the transmission frequency, G_T is transponder's antenna gain (in dBd) and G_R is reader's antenna gain (in dBi).

We also have,

$$a_F = 10log(\frac{P_{RO}}{P_{TI}}) \tag{2.3}$$

$$G_R = 10log(\frac{P_{RO}}{P_{RI}}) \tag{2.4}$$

where P_{RI} is power input at reader's antenna, P_{RO} is power output at reader's antenna and P_{TI} is power input at tag's antenna.

From (2.2), (2.3) and (2.4) we get,

$$10log(\frac{P_{RI} * 10^{G_R/10}}{P_{TI}}) = -147.6 + 20log(r) + 20log(F) - 10log(G_T) - 10log(G_R) \tag{2.5}$$

This can be transformed to derive the following expression for r.

$$r = \sqrt{\frac{P_{RI} * 10^{(G_R/10)} * 10^{14.76} * G_T * G_R}{P_{TI} * f^2}} \tag{2.6}$$

By substituting P_{TI} with the minimum power required as input at the tag's antenna, to make the tag operational, we obtain the reader antenna's range for a given power level. Even though the interrogation range of the reader's antenna can be increased by increasing the power level, a maximum is reached when the power reflected back by a just operational tag is just sufficient for the reader's antenna to detect the tag. On increasing the input power of reader's antenna a tag at a further distance might become operational but its response would not be detected by the reader's antenna. This maximum possible range can be calculated in the same fashion as equation (2.6)

$$r_{max} = \sqrt{\frac{(P_{TO})_{min} * 10^{14.76} * G_R * G_T}{(P_{RI})_{min} * f^2}} \tag{2.7}$$

where $(P_{TO})_{min}$ is minimum power output by a just operational tag and $(P_{RI})_{min}$ is minimum reflected power from tag required as input at reader's antenna for it to detect the tag.

RFIDPlanner has been designed following the above principles and keeping in mind the stated assumptions and design drivers. We now discuss the architecture and implementation of *RFIDPlanner* in the following chapter.

Chapter 3

RFIDPlanner Architecture and Implementation

The architecture of *RFIDPlanner* has been designed in such a way that *RFIDPlanner* can be used for any general scenario, given the obstacles conform to assumptions stated in Chapter 2. The architecture of *RFIDPlanner* is shown in Fig. 3.1. Some of the important aspects of the architecture are discussed below followed by implementation details.

3.1 Inputs

RFIDPlanner requires the floor plan of area to be covered and reader and tag specifications as inputs. A snapshot of the input GUI is shown in Fig. 3.2.

- **Floor Plan**: The floor plan of the area to be covered is in the form of an XML file. It provides dimensions of the area to be covered and details of obstacles present in the given area. For every obstacle, the XML file contains its type, thickness and end coordinates of the line segment passing through the center of the obstacle along its length. This information, along with attenuation values from Table 2.1, is subsequently used to determine attenuation at obstacles. Few sample floor plans are shown in Fig. 3.3.

- **Reader specifications**:

  - Antenna gain

  - Input power to reader's antenna

  - Operating frequency of antenna (in MHz)

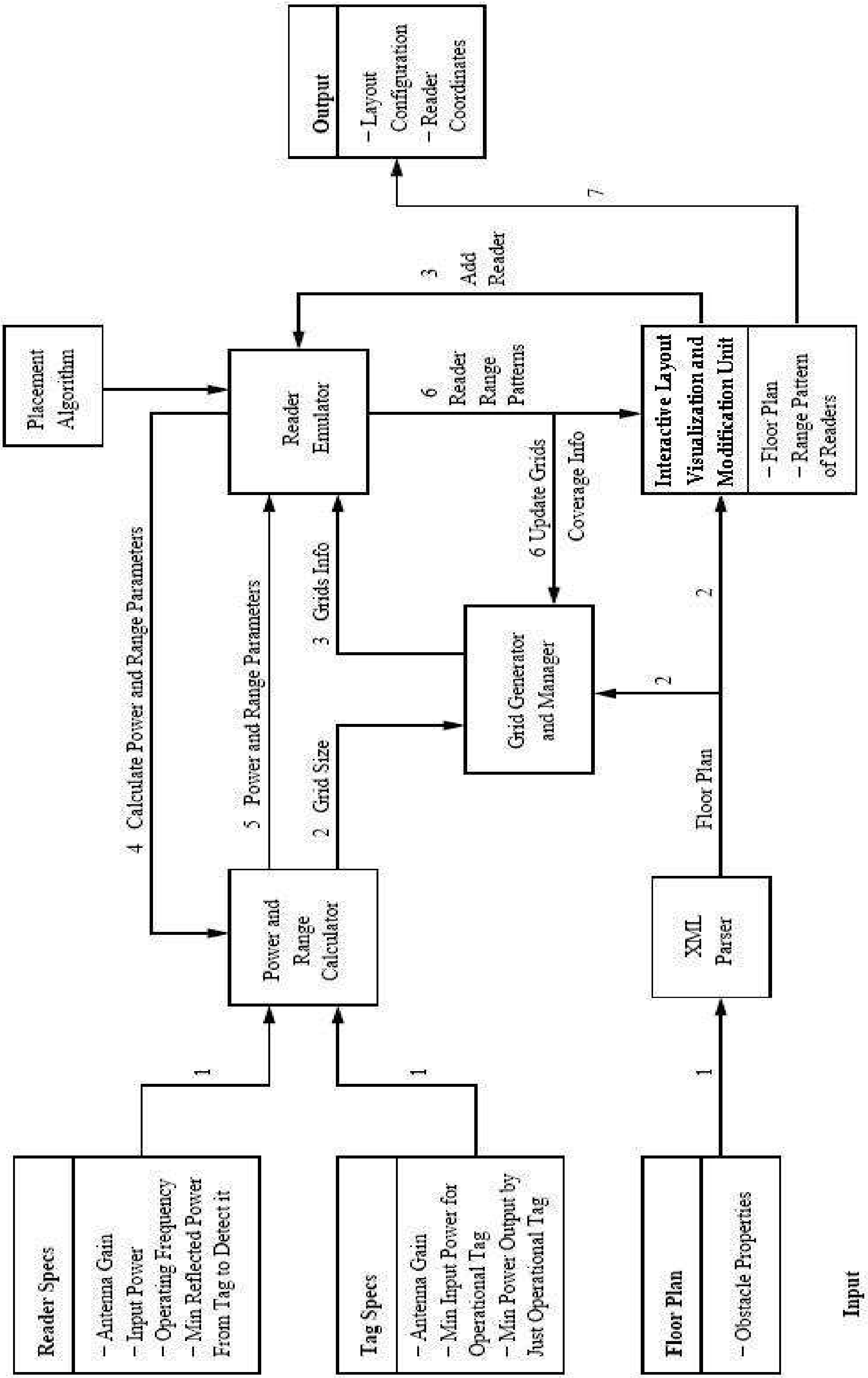


Figure 3.1: *RFIDPlanner* Architecture

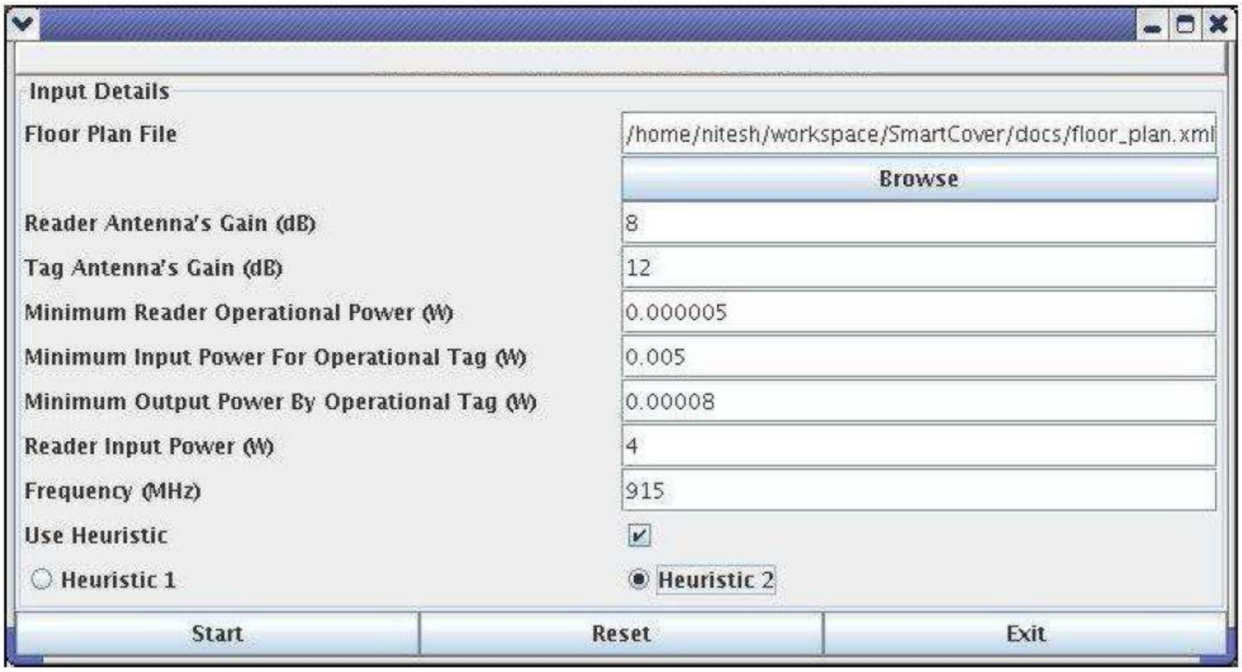


Figure 3.2: Snapshot of the input GUI

  - Minimum reflected power from a tag, required at reader's antenna for it to detect the tag

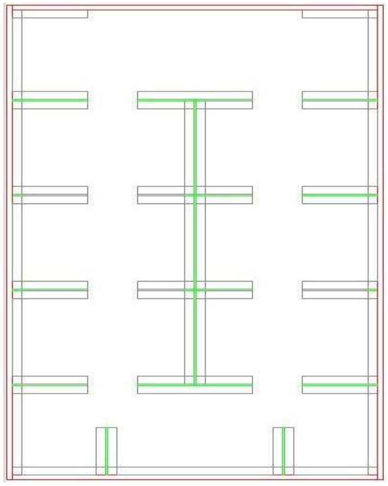
- **Tag specifications**:

  - Antenna gain
  - Minimum input power required at its antenna, for the tag to be operational
  - Minimum power output by a just operational tag

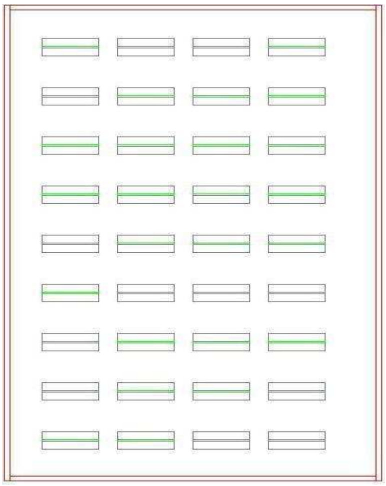
Reader and tag specifications, together, are required to calculate reader's range patterns.

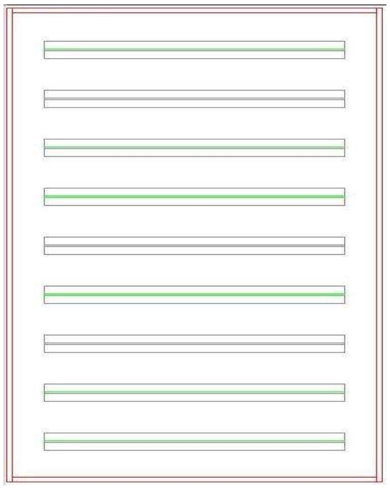
3.2 Power and Range Calculator

This module forms a repository of formulae used for calculating power attenuation levels as well as the reader range. The following set of formulae forms the current repository in this module.

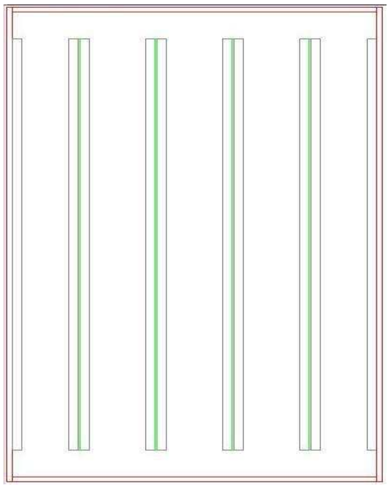


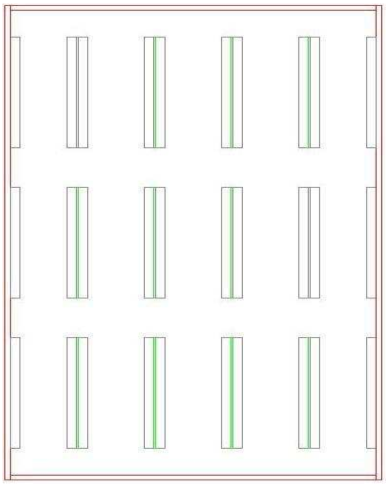
(a) Floor plan FP1





(b) Floor plan FP2 (c) Floor plan FP3





(d) Floor plan FP4 (e) Floor plan FP5

Figure 3.3: Floor Plan Visualizations

$$a_F = -147.6 + 20log(r) + 20log(F) - 10log(G_T) - 10log(G_R) \tag{3.1}$$

$$r = \sqrt{\frac{P_{RI} * 10^{(G_R/10)} * 10^{14.76} * G_T * G_R}{P_{TI} * f^2}} \tag{3.2}$$

$$r_{max} = \sqrt{\frac{(P_{TO})_{min} * 10^{14.76} * G_R * G_T}{(P_{RI})_{min} * f^2}} \tag{3.3}$$

All of these have been explained in detail earlier in Chapter 2.

The input values for reader and tag specifications are stored in this module. Once these input values have been read, the module calculates the reader range for the specified input power to the reader's antenna using equation 3.2. If this range is greater than the maximum reader range calculated using equation 3.3, the maximum reader range is set as the range for the given reader. This reader range is used to determine the grid size. The reader emulator module queries this module while emulating a reader's range.

Equation 3.1 is used for calculating the free space attenuation given the distance from the reader and vice versa.

This is an independent module which provides the flexibility of changing the formulae used for power and range calculations without effecting the operations of other modules. It also allows addition of new formulae in future which might be needed for any additional functionality.

3.3 XML Parser

The XML parser reads the input floor plan XML file and supplies the information to the gird generator module and display unit. Also, once the grids have been generated, the XML parser outputs an XML file containing the grid details. A snippet of the input file for floor plan FP1 is shown in Fig. 3.4.

3.4 Overlay Grid Structure

The *RFIDPlanner* Architecture is primarily based on an overlay grid structure. The area to be covered is divided into square grids of uniform size. The grid size is chosen such that the diagonal of the grid is equal to the maximum range of the readers to be deployed

```
<?xml version="1.0" ?>
<floorplan>
    <dimentions x="20" y="30"/>
    <obstacle>
        <number>1</number>
        <point x="0.15" y="0" z="0"/>
        <point x="0.15" y="30" z="0"/>
        <thickness>0.3</thickness>
        <type>Brick Wall</type>
    </obstacle>
    <obstacle>
        <number>2</number>
        <point x="19.85" y="0" z="0"/>
        <point x="19.85" y="30" z="0"/>
        <thickness>0.3</thickness>
        <type>Brick Wall</type>
    </obstacle>
                    ⋮
```

Figure 3.4: Input XML File for Floor Plan FP1

in the area. Fig. 3.5 shows how an overlay grid structure would look like on floorplan FP1.

Each grid contains details of sections of obstacles present in it. It also keeps track of readers covering it and the portion covered by that reader. While computing the reader range, we need to check for interference of radio waves with obstacles. Using the grid system ensures that we check for interference with only the obstacles present in the grids around, and including, the grid in which the reader is placed. Thus we avoid redundant checking of interference with far away obstacles. This makes the mathematical modeling of the effect of obstacles on reader range less computation intensive, and hence faster.

Given the floor plan and grid size, the *Grid Generator and Manager* generates the grid structure which is then used by the reader emulator while emulating reader ranges. It also provides the functionality of storing reader range information, generated by the reader emulator, in corresponding grids.

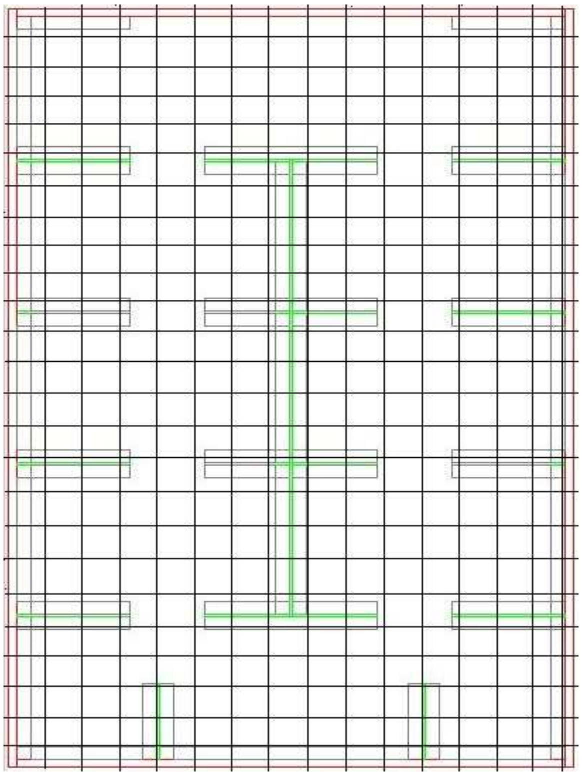


Figure 3.5: Overlay grid structure for Floor Plan FP1

3.5 Reader Emulator

Reader Emulator sits at the heart of *RFIDPlanner* performing the most important task of emulating attenuation patterns. While emulating a reader's range we set two parameters for emulating the range: *number of sections* and *number of attenuation levels*. These are described in more detail below followed by the algorithm used for calculating a reader's range attenuation patterns.

3.5.1 Number of Sections

The first parameter is the *number of sections* in which the circular range of the reader can be divided into, as shown in Fig. 3.6(a). Within each section, all points on the arc at a given distance from the center are assumed to receive the same level of power as the point on the center of that arc. In other words, it is assumed that all radio waves in a section will behave in the same manner as the radio wave passing through the center of that section.

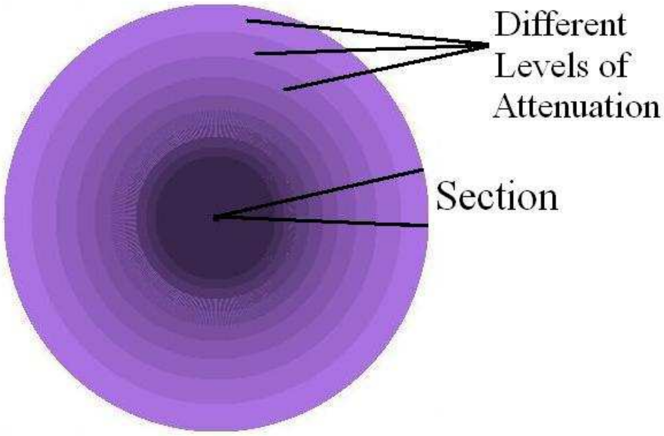
Thus, any obstacle that interferes with the radio wave passing through the center of a section, will be assumed to interfere with all the radio waves in that section. This makes approximating the range attenuation patterns easier. The accuracy of this approximation depends on the *number of sections* in which the range is divided. It was observed that dividing the range into 360 sections delivered quite accurate results. Increasing the number of sections beyond this unnecessarily increased the time for computation and also gave rise to computational errors due to the extremely small size of sections.

3.5.2 Number of Attenuation Levels

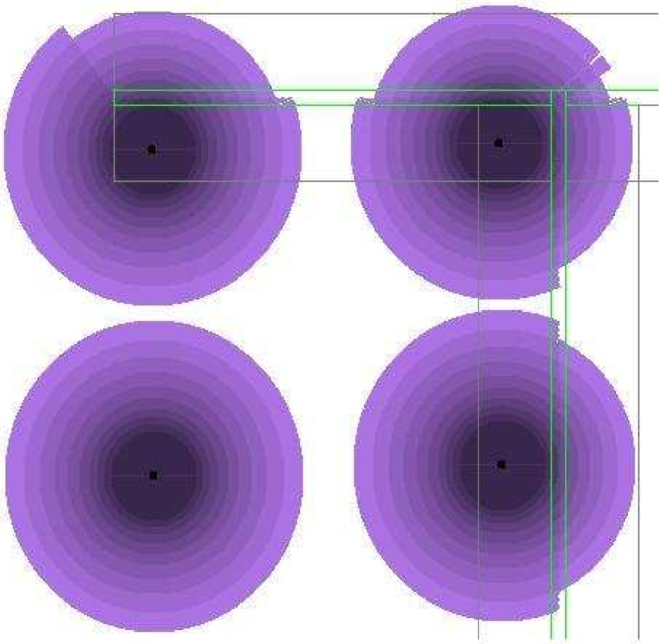
Each section is further divided into a *number of attenuation levels*, which forms the second parameter. Each level has a minimum and a maximum allowed attenuation of power received from the reader. For the first level, minimum attenuation is 0, which will be at the reader itself. For an intermediate level, minimum attenuation allowed is the same as the maximum attenuation allowed for the previous level. For the last level, maximum attenuation allowed is the total free space attenuation attained over the range of the reader in absence of any obstacles.

Each attenuation level is represented by a particular gradient of a color. As the level increases the color gradient assigned is lighter than the previous one. Thus, in the graphical visualization, gradients corresponding to the increasing attenuation levels give a fading effect to the reader's range, as shown in Fig. 3.6(a). This parameter is used to aid in visualizing range attenuation patterns. If an obstacle is present in the path of a radio wave, the attenuation in power of the radio wave just after the obstacle takes a sudden jump. This sudden increase in attenuation would be depicted clearly in the visualization as shown in Fig. 3.6(b). Dividing a section into 10 attenuation levels gives a fairly accurate approximation of range attenuation behavior.

For each section we maintain a list of end coordinates corresponding to attenuation levels. The radio wave passing through the center of the given section passes through these set of coordinates. The start coordinate of a level is the same as the end coordinate of the previous level. For the first level, the start coordinate is the position of the reader itself.



(a) Reader range pattern in the absence of any obstacles



(b) Range attenuation patterns in presence of obstacles

Figure 3.6: Range attenuation patterns

3.5.3 Reader Range Emulation Algorithm

While emulating reader range attenuation behavior for a particular attenuation level in a given section, we basically calculate the start and end coordinates of this attenuation level. As mentioned earlier, the start coordinate is the same as the end coordinate for previous level, hence there is no need to find it again.

Every attenuation level has a maximum allowed attenuation of power received from the reader at its end coordinate. This end coordinate would lie at the maximum distance from the reader in absence of any obstacle. If any obstacles are present in this or any previous attenuation level of the same section, the attenuation due to these obstacles will also have to be taken into consideration. Thus, the total free space attenuation would be reduced and the end coordinate would be shifted towards the reader. This would therefore result in reducing the effective range of the reader.

The main steps of the algorithm are described below. A more detailed version of this algorithm for calculating a reader's range pattern is shown in Fig. 3.7.

Let us consider the j^{th} attenuation level of the i^{th} section for a given reader. The end coordinate for the previous attenuation level has been calculated and obstacles, that may lie in that region, taken in to consideration.

1. Calculate the maximum attenuation allowed for the j^{th} attenuation level, taking into account the attenuation due to obstacles present in regions of previous attenuation levels of the same section.

2. Calculate the free space distance from the reader, corresponding to this maximum attenuation level and find the appropriate end coordinate. Note that any obstacles that may lie in the region of j^{th} attenuation level till now have not been taken into consideration.

3. Check if there are any obstacles that lie in the region of j^{th} attenuation level, *i.e.* between the start and end coordinates for this attenuation level.

4. If there are no such obstacles, then the coordinate calculated above is the required end coordinate.

5. If there are obstacles, consider the first obstacle from the start of this attenuation level. Calculate the total attenuation at this obstacle, which includes the free space

```
// maximum atttenuation of power allowed
maxAtten = GetMaxAttenuation();
attenInterval = maxAtten / noOfAttenLevels;
for(i=0; i< noOfSections; i++) {
        // total attenuation becasue of obstacles in this section. keeps adding up
        obsAtten = 0;
        start = GetReaderPosition();
        for(j=0; j<noOfAttenLevels; j++) {
                maxAttenForLevel = (j+1)*attenInterval − obsAtten;
                freeSpaceDist = GetFreeSpaceDist(maxAttenForLevel);
                end = FindCoordinate(freeSpaceDist, i);
                while(maxAttenForLevel > 0) {
                        obstacle = FindNextInterferingObstacle(i, start, end);
                        if(obstacle == null) {
                                break;
                        }
                        else {
                                dist = FindDistance(reader, obstacle);
                                freeSpaceAtten = GetFreeSpaceAtten(dist);
                                if(freeSpaceAtten + obstacle.getAtten() >= maxAttenForLevel) {
                                        end = FindInterferenceCoord(i, obstacle);
                                        break;
                                }
                                else {
                                        start = FindInterferenceCoord(i, obstacle);
                                        maxAttenForLevel −= obstacle.getAtten();
                                        freeSpaceDist = GetFreeSpaceDist(maxAttenForLevel);
                                        end = FindCoordinate(freeSpaceDist, i);
                                }
                                obsAtten += obstacle.getAtten();
                        }
                }
                AddLevel(i,j,end);
                start = end;
        }
}
```

Figure 3.7: Algorithm for reader range emulation

attenuation upto this obstacle and the attenuation due to all obstacles that lie between the reader and the current obstacle, including the current obstacle.

6. If this attenuation is greater than the maximum attenuation allowed for j^{th} attenuation level, then the required end coordinate for j^{th} attenuation level lies on this obstacle.

7. If the attenuation is less, the start coordinate is shifted to the point on the current obstacle and the entire procedure is repeated until the maximum allowed attenuation for j^{th} attenuation level is reached.

The above procedure is repeated for all the attenuation levels in all the sections.

A placement algorithm can be used to automatically generate appropriate reader positions, using the reader emulator, for completely covering the given area. Two such algorithms have been described in Chapter 4.

3.6 Interactive Layout Visualization and Modification Unit

The interactive layout visualization and modification unit forms another important part of *RFIDPlanner*. The visualization unit provides an interactive GUI which displays reader range patterns and the floor plan of the given scenario, showing distinct obstacles in different colors for easy visualization. Few sample floor plans are shown in Fig. 3.3. A zoom feature has been implemented to enhance visualization of reader range patterns.

The interactive modification unit is tied up with the reader emulator to perform the main tasks of adding new readers and moving or deleting existing ones. Users can add new readers at any location in the given area, simply by clicking on the desired location. This information is then passed on to the reader emulator. Once range attenuation patterns are calculated by the reader emulator, the reader and its range attenuation patterns are displayed as shown in Fig. 3.6(b).

Users can also move readers to other locations to observe any differences in the range attenuation patterns at the new and old reader positions. The entire process has been made interactive, allowing users to modify the reader layout during runtime.

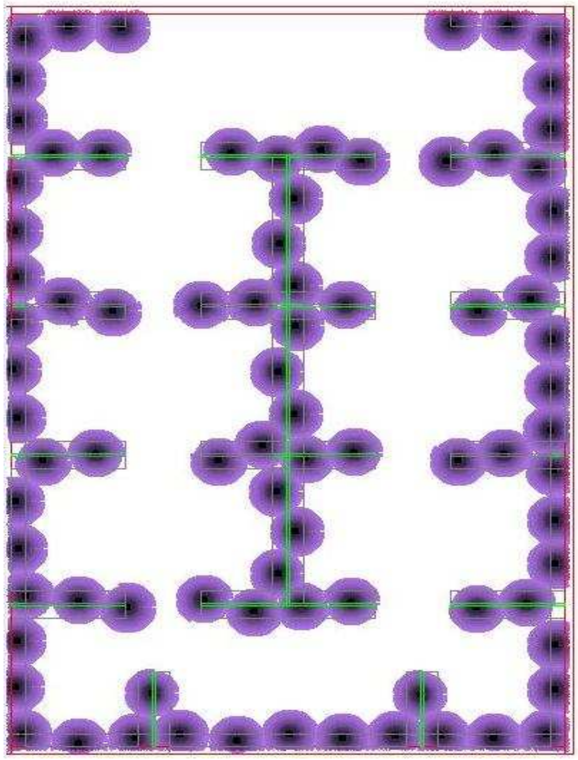


Figure 3.8: Sample Output for Floor Plan FP1

3.7 Output

The final output of *RFIDPlanner* is a visualization of:

1. Floor plan and layout configuration of RFID readers in the given area

2. Range attenuation patterns for RFID readers placed in the given area

A sample output for floorplan FP1 is shown in Fig. 3.8. The configuration of reader placement shown in Fig. 3.8 was done by manual placement of readers. Additional features to save the layout configuration information, including reader coordinates, and grid information in form of xml files can be easily added.

3.8 Architecture Flexibility

The *RFIDPlanner* architecture has been designed and developed in a flexible manner to allow changing configuration of different modules as well as addition of independent modules to enhance the functionality of the tool. For example, it is possible to build in support for different type of readers and tags available in the market. The database of different types of obstacle materials and their corresponding empirically determined propagation loss data can be easily expanded to include new information.

The architecture also allows addition of independent heuristics which can generate sample configurations of reader placement according to given scenarios. These configurations can then be visualized using *RFIDPlanner* and appropriately modified to suit requirements. The architecture thus allows *RFIDPlanner* to be easily adaptable to different user needs. In Chapter 4 we present a case study of a shopping mall scenario and two different heuristics, designed keeping in mind the requirements of a shopping mall, which have been implemented using *RFIDPlanner*.

3.9 Implementation

The *RFIDPlanner* architecture presented above has been implemented in Java. The high level design of *RFIDPlanner* is depicted in the class diagram shown in figure 3.9.

Once the inputs are read, maximum reader range is calculated and grids are generated. The interactive layout visualization and modification unit shows the floor plan to which the user can add readers at any location on the given area. The reader range attenuation patterns are then emulated and displayed on the floor plan.

`PowerAndRange` is used to store the reader and calculate the maximum reader range. `GenerateGrid` reads the floor plan XML file, stores the list of obstacles and generates the grids. `Coordinate`, `Obstacle` and `Grid` are the classes represent objects corresponding to their names.

Once the grids are generated, `DisplayGui` displays the floor plan. The user can now add new readers at any location in the given area by clicking on the desired location. A list of readers, added by the user, is maintained by `ReaderController`. When a reader is added, `Reader` calculates the attenuation patterns using grid details from `GenerateGrid`, mathematical modeling from `Geometry`, power and range formulae from `PowerAndRange` and empirical data regarding attenuation due to obstacles from `AttenuationValues`. The attenuation patterns are then used by `DisplayGui` to display the patterns to the user.

The code of *RFIDPlanner* has about 4700 lines of Java in 17 files containing a total of 17 different classes. The sample input XML files for the different floor plans used, contained between 150-850 lines.

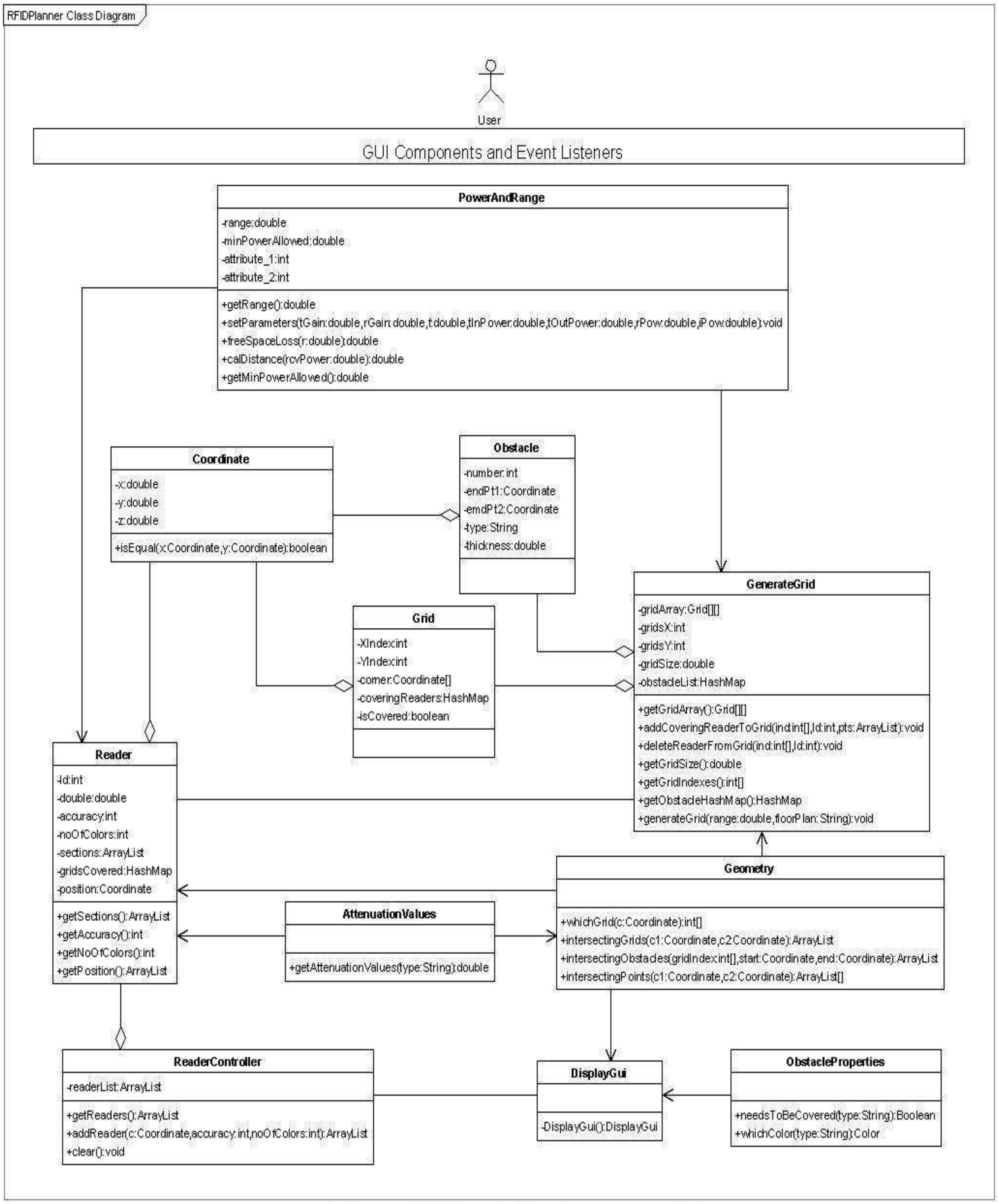


Figure 3.9: *RFIDPlanner* Class Diagram

In the following chapter we describe two heuristics that have been implemented, and evaluated, using *RFIDPlanner*.

Chapter 4

RFIDPlanner Evaluation

To evaluate *RFIDPlanner* we designed and implemented two heuristics for a shopping mall scenario. A shopping mall usually consists of a large hall, which has several wooden or plastic partitions for different items and brands. Each partition, in turn has several steel or glass shelves on which the items are displayed. In such a scenario, tagged items are confined to the shelves, and hence we need to ensure that all the shelves present in the given area are fully covered by the RFID reader network.

Assuming a similar given scenario, which has a large hall made of brick walls and consisting of several wooden partitions and steel shelves, we have designed two different heuristics to automatically generate reader placement configurations ensuring complete coverage of the area where tagged items will be placed *i.e.* shelves. Walls, partitions and shelves form the set of obstacles (with different propagation loss properties) that we will be dealing with. Thus all obstacles of type shelf will need to be fully covered. These heuristics are described in detail in the following sections.

4.1 Heuristic H1

Heuristic H1 has been designed with the aim of validating the basic functioning of *RFID-Planner* and of using it for comparison with other heuristics. It depends largely on different modules of *RFIDPlanner* such as the overlay grid structure and reader emulator to come up with placements of readers ensuring complete coverage of the desired area. It is described in detail below.

1. H1 starts by selecting the first grid that contains an edge (along the breadth) of a shelf and placing a reader on the middle of this edge.

```
gridsToBeCovered = FindGridsContainingShelves();
while(gridsToBeCovered != null) {
    firstUncoveredGrid = FindFirstUncoveredGridContainingShelfEdge();
    reader = placeReader(firstUncoveredGrid);
    partiallyCoveredGrids = FindPartiallyCoveredGridsContainingShelves(reader);
    while(partiallyCoveredGrids != null) {
        newReader = placeReader(partiallyCoveredGrids.get(0));
        newPartiallyCoveredGrids = FindPartiallyCoveredGridsContainingShelves(newReader);
        partiallyCoverdGrids.add(newPartiallyCoveredGrids);
        girdsToBeCovered.remove(partiallyCoveredGrids.get(0));
        partiallyCoveredGrids.remove(0);
    }
}
```

Figure 4.1: Algorithm for Heuristic H1

2. The *RFIDPlanner* reader emulator is then used to emulate the range pattern and determine which all grids are partially or fully covered by this reader.

3. The partially covered grids are added to a global FIFO list. These partially covered grids are then analyzed one by one.

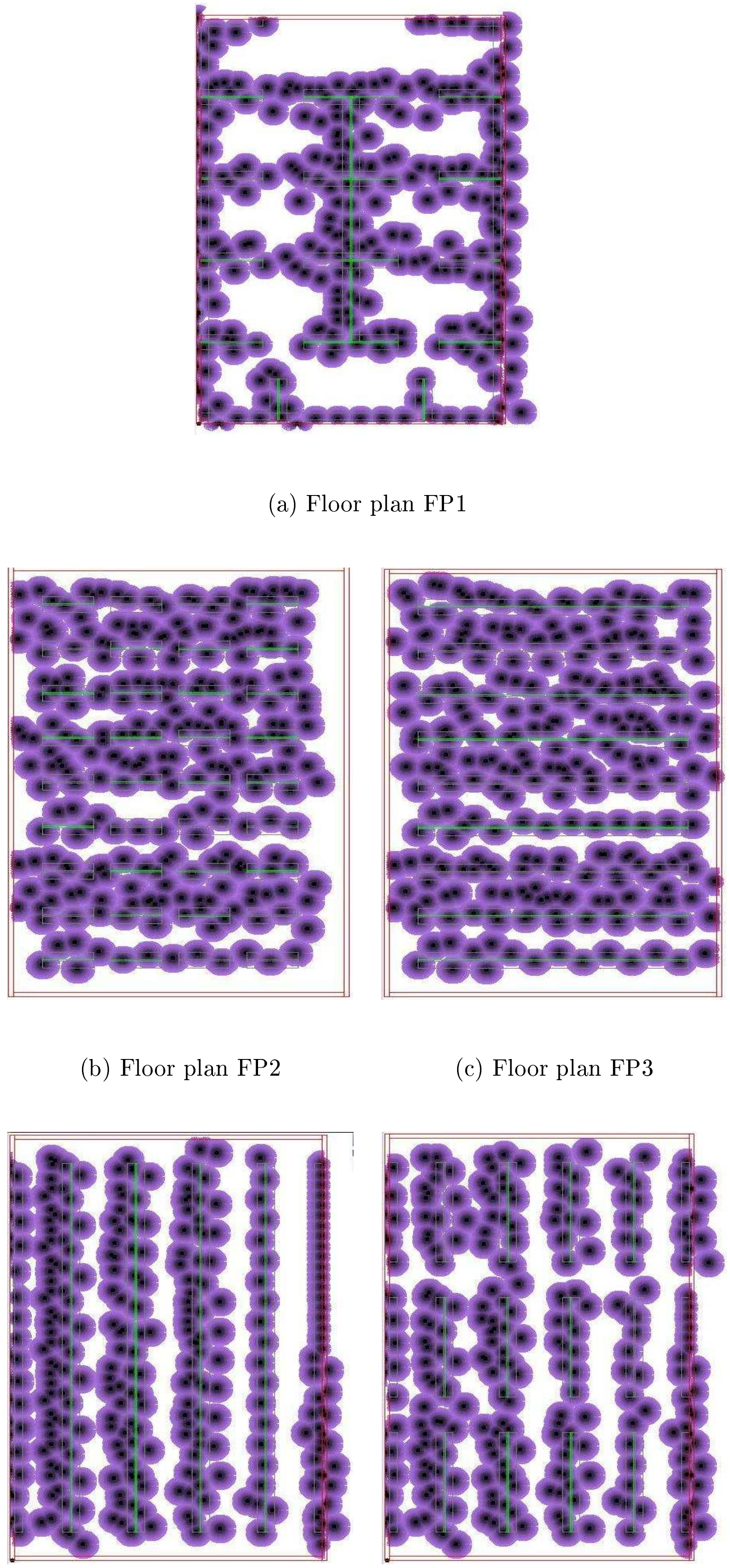
4. If a partially covered grid does not contain any shelf, no readers are added to further cover the grid. Note that we are only interested in covering the shelves, hence no readers are added.

5. If there are shelves in this grid, another reader is placed so as to cover the entire grid while trying to minimize interference with readers already covering the grid.

6. The new grids which are now partially covered by the new reader would be added to the FIFO list and the procedure is repeated until all the shelves are covered completely.

The algorithm for H1 is shown in Fig. 4.1. Sample configurations of reader placement in the floor plans shown in Fig. 3.3, generated using H1 are shown in Fig. 4.2.

Apart from validating the basic functioning of *RFIDPlanner*, this heuristic also demonstrates the ease with which underlying modules of *RFIDPlanner* can be accessed and utilized by heuristics which might be added later to *RFIDPlanner*.



(a) Floor plan FP1

(b) Floor plan FP2

(c) Floor plan FP3

(d) Floor plan FP4

(e) Floor plan FP5

Figure 4.2: Readers placement configurations using Heuristic H1

4.2 Heuristic H2

The design of heuristic H2 is an attempt to capture the logic we would normally use while placing RFID readers manually in the given scenario. For the sake of algorithmic convenience we modify the floor plan such that if any two shelves are touching each other, we insert an imaginary obstacle with zero thickness and zero propagation loss, between the two.

The algorithm for H2 is shown in Fig. 4.3. The following objectives were kept in mind while designing H2.

- Ensure that there is no need for creation of additional infrastructure for deploying readers. In other words, it should be made sure that the readers are placed on the existing obstacles.
- If there are adjacent shelves, we try to place a RFID reader such that it covers appropriate portions of both the shelves, thus minimizing the number of RFID readers required.

To achieve these objectives, instead of considering all the grids containing shelves as was done in H1, we directly make a list of all the shelves. These shelves are now analyzed one by one. The following steps are then taken to cover the shelves.

1. If a shelf is not covered, we look for adjacent shelves to the given shelf.
2. If uncovered adjacent shelves are not found, we simply place RFID readers along the axis of the shelf (the line running through the center of the shelf along its length) ensuring minimum interference with earlier placed reader, while covering the entire shelf.
3. In case an uncovered adjacent shelf is found, we first check for the effect of propagation loss due to the obstacle between the given shelf and the adjacent shelf. In this case a RFID reader is placed next to the obstacle, on the given shelf.
4. If the effective interrogation range is less than the combined thickness of the obstacle and the adjacent shelf, then it is not possible for the RFID reader to cover parallel portions of the shelf and the adjacent shelf simultaneously. Thus we place the RFID reader along the axis of the self and ignore its coverage effect on the adjacent shelf.

```
shelves = GetListOfShelves();
for(i=0; i<shelves.size(); i++) {
        shelf = shelves.get(i);
        if( !isCovered(shelf)) {
                adjacentShelf = FindAdjacentShelf(shleves.get(i));
                obstacle = ObstacleBetween(shelf, adjacentShelf);
                if((adjacentShelf== null) || isCovered(adjacentShelf)) {
                        placeReadersAlongAxis(shelf);
                }
                else if(attenuatedReaderRange(shelf, adjacentShelf) <
                                        adjacentShelf.thickness + obstacle.thickness()) {
                        placeReadersAlongAxis(shelf);
                }
                else {

                        placeReadersAlternatelyAlongObstacle(shelf,adjacentShelf,Obstacle);
                }
        }
}
```

Figure 4.3: Algorithm for Heuristic H2

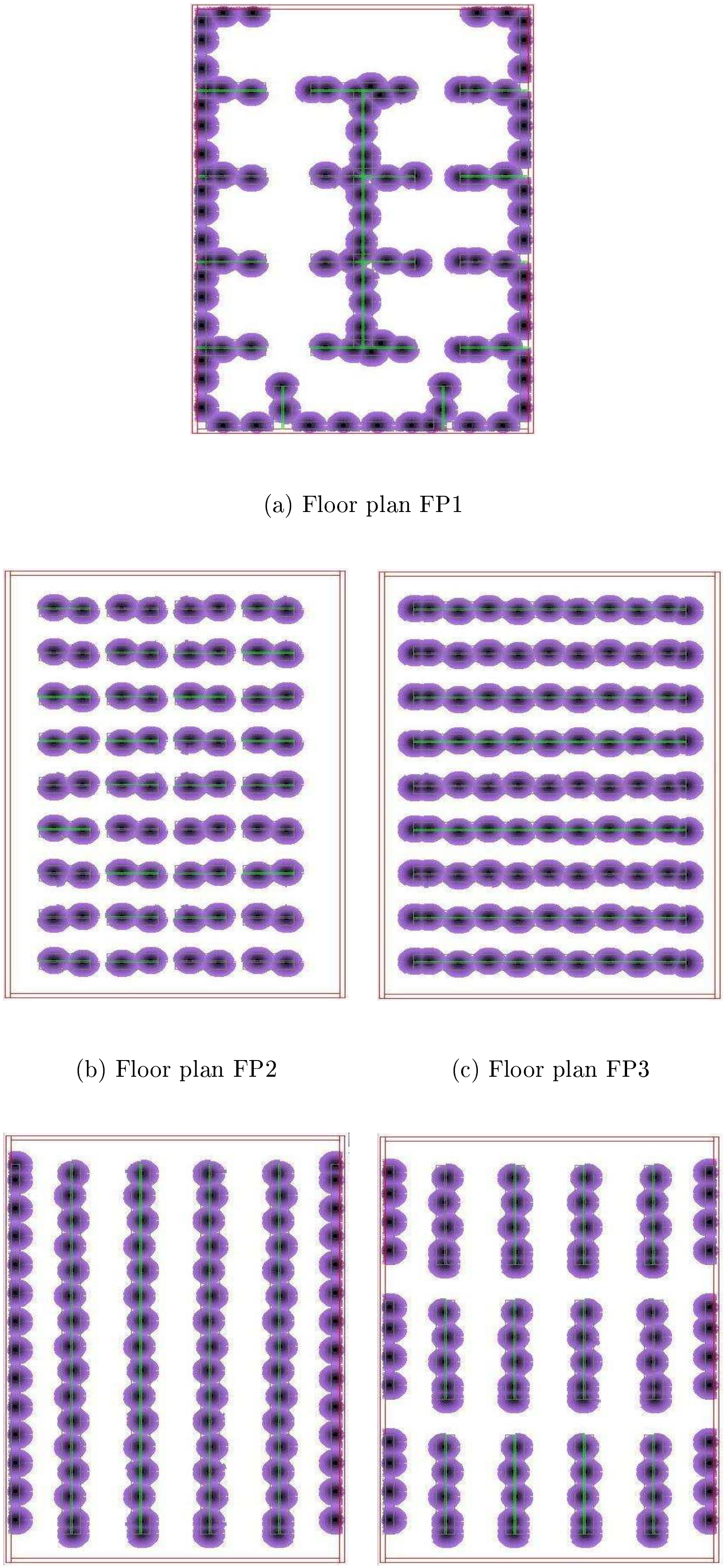
5. If the interrogation range is greater than the combined thickness of the obstacle and the adjacent shelf, we place the RFID reader next to this obstacle on the given shelf.

6. The next RFID reader is placed again next to the same obstacle but this time on the adjacent shelf, such that the interference is minimum between the ranges of the two RFID readers. Readers are thus placed in an alternate manner on the shelf and the adjacent shelf, aligned to the obstacle between the two, till atleast one of them is fully covered.

This procedure is repeated for all the shelves, thus ensuring that all the desired area is fully covered.

Sample reader placement configurations for the floor plans shown in Fig. 3.3, generated using H2 are shown in Fig. 4.4.

H2 makes use of the overlay grid structure of *RFIDPlanner* to determine the adjacent obstacles and adjacent shelves. It also uses the module used for propagation loss and range calculations.



(a) Floor plan FP1

(b) Floor plan FP2 (c) Floor plan FP3

(d) Floor plan FP4 (e) Floor plan FP5

Figure 4.4: Readers placement configurations using Heuristic H2

Table 4.1: Deployment Results

Floor Plan	**No. of RFID Readers Placed using**		
	Manual Placement (MP)	**Heuristic H1**	**Heuristic H2**
FP1	85	317	102
FP2	**72**	262	**72**
FP3	77	285	99
FP4	78	312	92
FP5	72	289	84

4.3 Performance Evaluation

It has been shown that coming up with an optimum configuration for coverage planning is an NP hard problem [6]. Thus, a good way to judge the performance of a heuristic implemented in *RFIDPlanner* would be to compare the number of RFID readers placed by the heuristic to cover the entire given area with the number of RFID readers that would be placed manually. We assume that while placing RFID readers, the user would use proper discretion to avoid redundancy and interference. The results for deployment of RFID readers in the floor plans shown in Fig. 3.3 are shown in Table 4.1. A comparison of these results is made in Table 4.2.

From the results it is clear that H1, which is a naive heuristic developed mainly for the purpose of validation, came up with reader placement configurations containing approximately four times the number of readers a user would have manually placed.

On the other hand, H2, which follows a more practical approach, performed much better and generated configurations containing approximately 1.2 times the readers placed manually. The configurations generated by H2 are also quite similar to the configurations got by manual placement of RFID readers. It is interesting to observe that for floor plan FP2, the number of readers placed by H2 was *exactly* the same as the number of readers placed manually. This demonstrates that H2 has been successful, to a large extent, in its objective of capturing the logic that would be used during manual placement of readers.

Table 4.2: Comparisons of Results

Floor Plan	**% of No. of RFID Readers**			
	H1/MP	**H2/MP**	**H1/H2**	**H2/H1**
FP1	373	120	311	32
FP2	364	**100**	364	27
FP3	370	129	288	35
FP4	400	118	339	29
FP5	401	117	344	29

4.4 Algorithmic Complexity

4.4.1 Complexity of H1

For the algorithm for H1, shown in Fig. 4.1, in the *worst case scenario*, all the shelf containing uncovered grids will be scattered away from each other, *i.e.* when a reader is placed to cover one of these grids, the partially covered grids thus generated will not contain any shelves. In this case the function $FindFirstUncoveredGridContainingShelfEdge()$ will be called for each grid containing a shelf. This function has a complexity of $O(n)$, where n is the total number of grids. The complexity for placing a reader is $O(1)$. A reader can at a time cover a maximum of 14 grids. Therefore, the complexity for finding partially covered grids containing shelves is $O(14)$. Thus, the total complexity in this case will be $O(n(n+1+14))$, or $O(n^2)$.

In the *best case scenario*, all the grids containing shelves will be nearby. Here, when a reader is placed in one of the grids, some or all of the resulting partially covered grids will contain shelves. Readers will be placed to cover these grids, which would intern generate more such shelf containing partially covered grids and so on until all the required grids are covered. In this case, the function $FindFirstUncoveredGridContainingShelfEdge()$, with complexity $O(n)$ is called only once. For covering each partially covered grid, complexity for placing a reader is $O(1)$ and for finding partially covered grids containing shelves is $O(14)$. Therefore, complexity for covering all partially covered grids is $O(n(1+14))$, $i.e. O(n)$. Thus, complexity for the complete algorithm is $O(n+n)$, $i.e. O(n)$.

4.4.2 Complexity of H2

In the algorithm for H2, shown in Fig. 4.3, each shelf is analyzed once. The complexity for finding the adjacent shelf is $O(m)$, where m is the total number of shelves in the given area. Once an adjacent shelf has been found, the complexity for finding the obstacle between the two shelves is $O(1)$. The complexity for placing readers, either along the axis of the shelf or along the obstacle, is $O(l)$ where l is the length of the shelf. Thus, the overall complexity becomes $O(m(m + l))$. If we consider the length of shelves to be constant, the complexity becomes $O(m^2)$.

The ease of integration of the two heuristics in *RFIDPlanner* illustrates the flexibility of *RFIDPlanner* which is inherent in its architecture. The performance of the heuristics using *RFIDPlanner* validate its functioning for different kinds of scenarios.

In the following chapter we discuss about a RFID middleware and the possibility of integrating *RFIDPlanner* in it.

Chapter 5

PINES^{TM} : RFID Middleware

Persistware for Internal Network EPC Solutions (PINES^{TM}) is an RFID middleware application developed by Persistent Systems Pvt. Ltd. (PSPL). PINES^{TM} integrates internal RFID networks of enterprises with conventional middleware such as internal ERP applications, providing features such as layout and device management frameworks, data aggregation and filtering services, query processing and event based notifications for enhanced decision making.

5.1 PINES^{TM} Framework

The PINES^{TM} framework is made up of several interrelated engines managing different aspects of the middleware and emulator. The main components of the framework are:

- Data collection and device management engine
- Layout management engine
- PML server
- Event store
- Decision support engine
- Automated actuation engine

These engines can be broadly grouped into three types.

1. Management engines, including layout and device management, which take care of representing and managing real or emulated locations, readers and antennas as well as movement of tags between antennas.

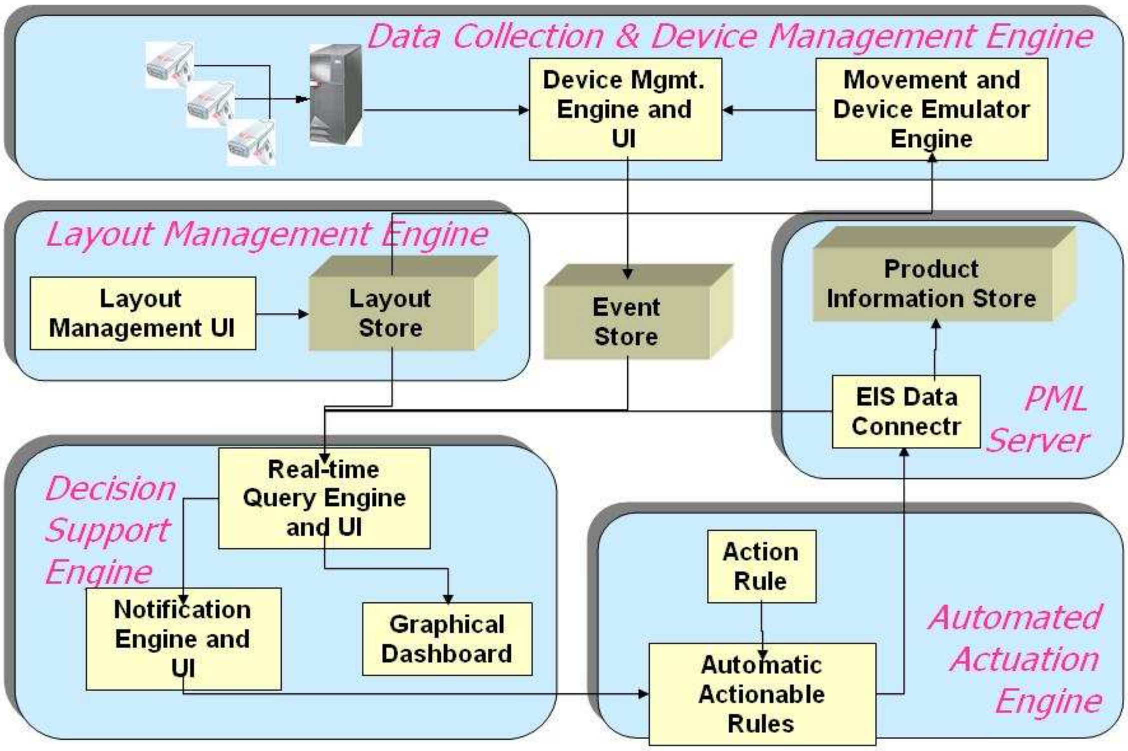


Figure 5.1: PINES^{TM} Framework

2. Information and data servers, including PML server and event store, which handle information regarding products in circulation and events associated to these.

3. Query processing and notification engines, including decision support and automated actuation engines, which are responsible for aiding in making real-time decisions based on predefined business rules.

Fig. 5.1 presents an overview of the structure of the framework, including the subcomponents of each engine.

5.2 *RFIDPlanner* and PINES^{TM}

For layout management the PINES^{TM} module currently uses a layout map containing information about different locations and reader placements, which has to be provided by the user. The reader layout configuration is also decided by the user, which might not be an optimized configuration. Also it would require the user to have knowledge about RFID readers and antennas for generating the required reader layout configuration to ensure adequate coverage.

RFIDPlanner can be integrated in the management engines to provide additional functionality to the PINES$^{\text{TM}}$ layout management engine during the initial phase of RF sensor network configuration. This can be done as follows.

1. Specifications of different tags and readers supported by PINES$^{\text{TM}}$ can be stored in a database. These specifications can directly be used as input for *RFIDPlanner*, thus eliminating the need for users to be familiar with this data.

2. As an input to the Layout Management Engine, take as input an XML file containing details of the locations that need to be covered.

3. This XML file is used as an input to *RFIDPlanner* which suggests a reader layout configuration using an appropriate heuristics. More heuristics can be added to *RFIDPlanner* to cover several different type of scenarios.

4. The suggested reader layout configuration is displayed using visualization and modification unit of *RFIDPlanner*, which can be embedded in the Layout Management User Interface of PINES$^{\text{TM}}$.

5. An option of applying different heuristics and comparing the configurations can be easily built in, as an additional functionality. The user can be given the option of choosing any of these configurations.

6. The chosen configuration can then be modified according to specific requirements by a user during runtime.

7. The final reader layout configuration is then saved in a format that can be directly used as an input for the Layout Management Engine.

Thus using *RFIDPlanner* along with PINES$^{\text{TM}}$ would help save user time by assisting in coverage planning and make PINES$^{\text{TM}}$ more user friendly.

Chapter 6

Conclusions

We explored the need for RFID domain specific coverage planning tools. We described in detail the design and architecture of *RFIDPlanner* - a coverage planning tool for RFID networks, including the algorithm used for emulating RFID reader range patterns. Further, the flexibility of its architecture was demonstrated by utilizing different modules of *RFIDPlanner* to develop two different heuristics for coming up with RFID readers placement configurations. The algorithms for these heuristics were explained in detail and result of their implementations were compared.

The *RFIDPlanner* architecture provides a basic framework for a RFID coverage planning tool. It presents a lot of flexibility for further addition of features, such as support for slant obstacles, new heuristics for different scenarios and direct support for different RFID readers and tags available in the market eliminating the need for the end user to be familiar with RFID reader and tag specifications. With such functionality enhancements, the scope for application of *RFIDPlanner* can be increased and thus utilized for coverage planning in much complex scenarios.

Given the high level of integrability that *RFIDPlanner* offers because of its flexible architecture, *RFIDPlanner* can also be integrated in a RFID middleware to provide a single platform for performing tasks ranging from coverage planning for RFID systems to deploying, monitoring and operating these systems.

Bibliography

[1] S. Iyer, "RFID: Technology and Applications", Kanwal Rekhi School of Information Technology, Indian Institute of Technology - Bombay, 2004. http://www.it.iitb.ac.in/ sri/talks/RFID-05.ppt

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

[3] S. Zvanovec, P. Pechac and M. Klepal, "Wireless LAN Networks Design: Site Survey or Propagation Modeling?", *Radioengineering, Vol. 12, No.4*, December 2003.

[4] T. Rappaport, "Wireless Communications: Principles and Practice", *Prentice Hall*, 1996.

[5] B. Silberberg, "Technical State of 868-870 MHz Radio Modules in the SRD Band", *AUBE'01, 12th International Conference On Automatic Fire Detection*, March 25 - 28, 2001. http://fire.nist.gov/bfrlpubs/fire01/PDF/f01063.pdf

[6] S. Anusha and S. Iyer, "RFIDcover: A coverage planning tool for RFID networks with mobile readers", *1st International Workshop on RFID and Ubiquitous Sensor Networks (USN)*, Nagasaki, Japan, Dec 2005.

[7] S.J. Fortune, D.H. Gay, B.W. Kernighan, O. Landron, R.A Valenzuela, and M. H. Wright, "WiSE Design of Indoor Wireless Systems: Practical Computation and Optimization", *IEEE Comput. Sci. Eng., vol. 2, pp. 58–68*, March 1995.

[8] SpectraGuard Planner, *AirTight Networks.* http://www.airtightnetworks.net/productsandservices/sg_planner.html

Acknowledgments

I express my sincere gratitude towards my guide **Prof. Sridhar Iyer** for providing me an opportunity to work in this exciting field and 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 **Swapnil Paranjpe** and **Rahul Sachdev** for their valuable guidance and support during the course of this project.

I would like to thank **Nitesh Dixit** and **Naval Bhandari** for their valuable suggestions and helpful discussions during the course of the project work.

Last but not the least, I would like to thank **Persistent Systems Private Limited** for sponsoring this project.

Nitesh Gupta

I. I. T. Bombay

May 17^{th}, 2006