“RESOURCE ALLOCATION ALGORITHM

USING DIRECTIONAL ANTENNAS IN WIMAX”

by

Neeraj Gurdasani

A project submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November, 2009

APPROVAL

of a project submitted by

Neeraj Gurdasani

This project has been read by each member of the project committee and has been found to be satisfactory regarding content, English usage, format, citation, bibliographic style, and consistency, and is ready for submission to the Division of Graduate Education.

Dr. Brendan Mumey

Approved for the Department of Computer Science

Dr. John Paxton

Approved for the Division of Graduate Education

Dr. Carl A. Fox

STATEMENT OF PERMISSION TO USE

In presenting this project in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this project by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this project in whole or in parts may be granted only by the copyright holder.

Neeraj Gurdasani

November, 2009

ACKNOWLEDGEMENTS

I am grateful to acknowledge and thank all of those who have assisted me in my graduate program at Montana State University. First, I would like to thank Dr. Brendan Mumey, my academic advisor who has been a huge inspiration for me here at MSU. Without his valuable guidance and quick feedbacks, this project would not have been possible. I take this opportunity to convey my deepest thanks to Dr. Richard S Wolff of Electrical and Computer Engineering department. His constant encouragement and support has made my stay here a memorable one. He has been a huge inspiration for me and over the past 2 years and I have learnt a lot under him. Special thanks are extended to Dr. Jian Tang for his regular inputs and prompt feedbacks.

I also want to thank project mate and friend Vishwanath Annavarapu for his guidance in this work. Finally, I would like to thank my parents and sisters for their unconditional support throughout my life.

TABLE OF CONTENTS

TABLE OF CONTENTS-CONTINUED

LIST OF FIGURES

LIST OF FIGURES-CONTINUED

ABSTRACT

This paper discusses different algorithms to create directional antenna patterns in OPNET Modeler. We describe algorithms to create conical and pencil beam antenna patterns. We also present a version of this algorithm that creates antenna patterns whose gain follows a cosine squared curve. We have also implemented tracking in OPNET to utilize these directional antenna patterns. These algorithms can be used to create antenna patterns for any wireless technology in OPNET such as WiFi and WiMAX.

This paper also discusses an integer linear programming (ILP) formulation for a resource allocation problem. This is a centralized optimization algorithm where the base station decides which subscriber station will be assigned to which relay station in the network such that the throughput of the entire network is maximized. We present some simulation results showing 10Mbps improved throughput using directional antennas with the assignment strategy. This is a centralized algorithm and can be used in any technology that has a centralized resource allocation scheme such as WiMAX.

CHAPTER 1

INTRODUCTION

Worldwide Interoperability for Microwave Access, more commonly known as WiMAX is a new telecommunications technology that provides wireless transmission of data using a variety of transmission modes, from point-to-multipoint links to portable and fully mobile internet access. WiMAX, which is standardized by IEEE in the 806.16 family, was first said to be able to run at frequencies up to 66 GHz. Because of the limitations of silicon hardware, at such high frequencies we have to change our hardware to Gallium Arsenide (GaAs). But GaAs is not as cheap as silicon. So for this and other reasons, WiMAX is standardized to run at 2-11 GHz.

IEEE 806.16 standard is a huge standard. Unlike IEEE 802.11, in which there is little room for experiments for a vendor, in IEEE 802.16 vendors have a lot of options, such as what frequency to run at, etc. This leads to a problem of coordination. For example a WiMAX Base Station made by Vendor-A which is made in China may not be compatible to run with WiMAX Subscriber Station made by Vendor-B in the US. To help in this coordination, WiMAX Forum was created. The WiMAX Forum is an industry led, not-for-profit organization formed to certify and promote the compatibility and interoperability of broadband wireless products based upon the harmonized IEEE 802.16/ETSI HiperMAN standard. So, a WiMAX base station which follows the WiMAX Forum spec will be compactable to run with a WiMAX Subscriber Station follows the WiMAX Forum.

Figure 1.1 - WiMAX Technology usages worldwide [Red–806.16d, Blue–802.16e]

WiMAX Technology has a lot of promise because it provides broadband services to wide area networks. Assuming a 14MHz channel and 5 bps/Hz, WiMAX can provide Data Rate up to 70Mbps[1]. 802.16 supports both OFDM and OFDMA frequency allocation methods for non-line of sight application. 802.16 supports transport layer protocols such as IPv4, IPv6, ATM, Ethernet, etc. The technology also supports adaptive antennas and space time coding. WiMAX can provide broadband services over a wide area network.

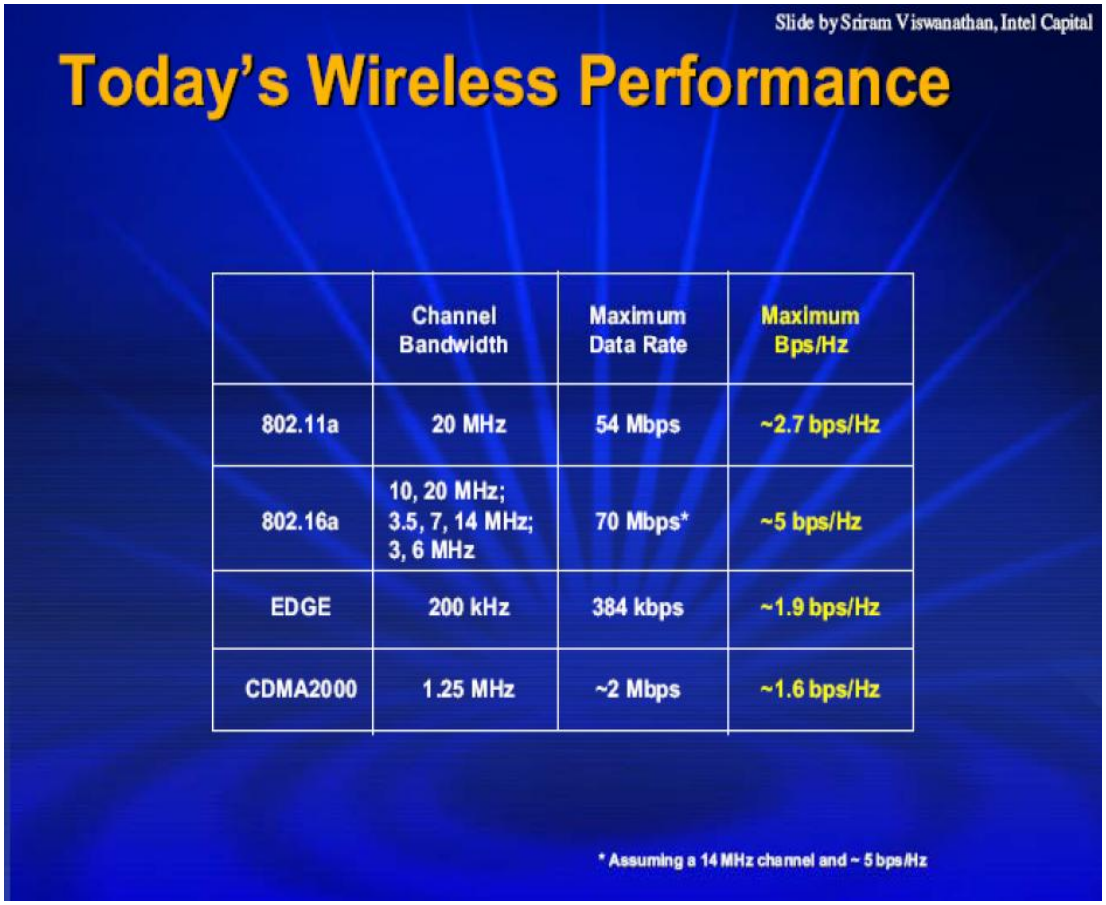

Figure 1.2 - Performance Comparison of today's Wireless Technologies

Directional Antennas

An antenna gives the wireless system three fundamental properties: gain, direction and polarization. Gain is a measure of increase in power. Gain is the amount of increase in energy that an antenna adds to a radio frequency (RF) signal. Direction is the shape of the transmission pattern. As the gain of a directional antenna increases, the angle of radiation usually decreases. This provides a greater coverage distance, but with a reduced coverage angle. The coverage area or radiation pattern is measured in degrees. These angles are measured in degrees and are called beamwidths[2].

A directional antenna or beam antenna is an antenna which radiates greater power in one or more directions as compared to an omni-directional antenna that radiates equal power in all directions around the antenna's vertical axis. By using directional antenna, we focus our radiated power towards the intended receiver, hence saving power by not radiating in all directions, allowing for increased performance on transmit and receive. The main Advantages of a directional antenna are

- Less interference[2]
- Higher gain[2]
- Higher adaptive modulation coding(AMC)[3]

Figure 1.3 - A directional antenna created by our project team at Montana State Univ.

Less Interference

An omni-directional antenna is an antenna that equally transmits (or receives) electromagnetic radiation to/from any arbitrary direction. In other words, it has a directivity of 0 dBi. The downside of an omni-directional antenna is that the antenna transmits information in the direction with no intended receiver. Hence, it is wasting

power in that direction. Moreover other nodes apart from the transmitter with the omni-directional antenna and the intended receiver, which are operating in the same frequency range, will get affected by the waves of the omni-directional antenna of the transmitter. So for these nodes, the signal to noise ratio (SNR) at the receiver will get adversely affected by this interfering wave. Hence, if we have a network with omni-directional antennas, the range of the transmitter will be significantly reduced.

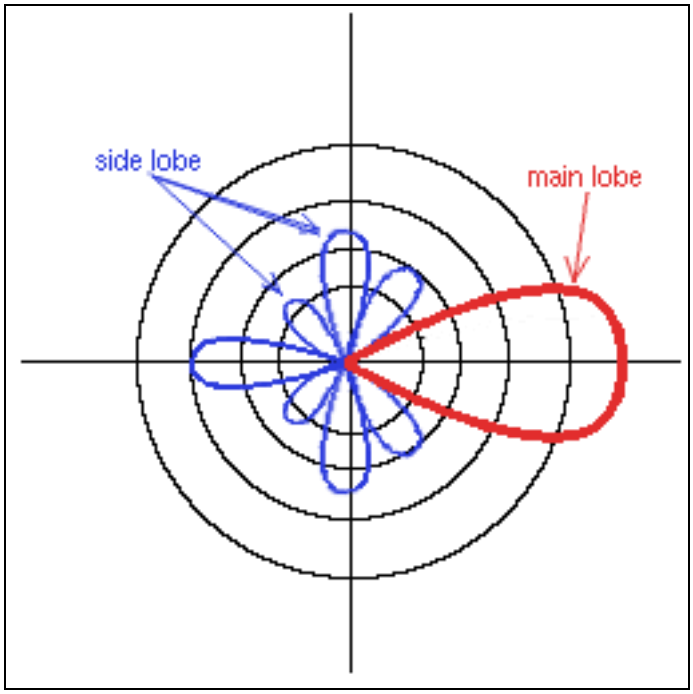

Figure 1.4 - Directional Antenna

On the other hand, if we use a directional antenna, as we can see from figure 1.4, the radiation in unwanted direction is limited to side lobes. Side lobes generally have a smaller gain as compared to the main lobe. Hence directional antennas have a smaller interference range. Hence, a network with nodes having directional antenna has much less interference as compared to a network with omni-directional antenna.

Higher Gain

We can say that an antenna amplifies the transmitting signal and sends it to space. Antennas have fixed amount of power to amplify the signal. Directional antennas radiate only in the directional of the intended user. Hence, they can use all their available power to amplify the signal in the direction of the receiver. In an onmi-directional antenna, all the available power is used to amplify the signal in all the directions. Hence, the main lobe of the directional antenna will have higher gain as compared that of an omni-directional antenna. Hence, the SNR at the receiver will be higher in case of the directional antenna as compared to the omni-directional antenna. As we can see in Figure 1.5, the main lobe of the directional antenna has higher gain as compared to omni-directional antennas.

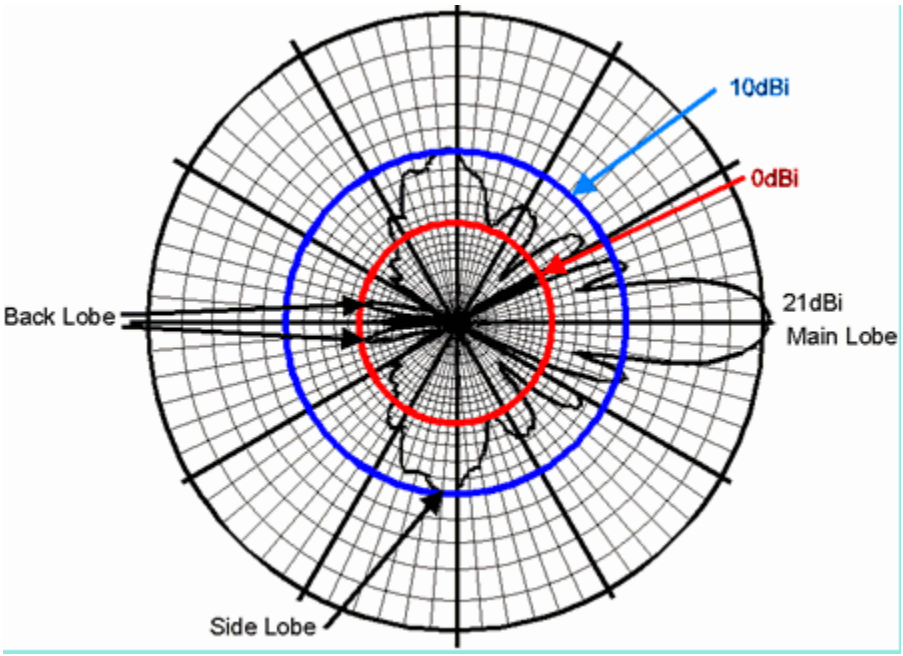

Figure 1.5 – Directional Antenna vs. Omni-directional Antenna

For a fixed power, let G_o be the gain of the omni-directional antenna and let G_d be the gain of the directional antenna. Let θ be the angle of the main lobe in the azimuth

plane and let Φ be the angle of the main lobe in the elevation plane for the directional antenna. Then, we can say that

$$G_d \propto \frac{1}{\theta * \phi} G_o$$

Equation 1.1 – Relationship between Gain of directional and omni-directional antenna [3]

Table 1.1 – Relationship between G_d , G_o , θ and Φ

θ and Φ (θ = Φ)	G_d
360	2 dBi ($G_d = G_o$)
180	6.33 dBi
90	14.03 dBi
45	20.05 dBi
30	23.58 dBi
15	29.6 dBi

Table 1.1 will show how G_d increases as θ and Φ decrease. In this table the value of G_o is 2dBi.

Higher Adaptive Modulation Coding

In wireless communication technologies such as WiFi and WiMAX, information bits are generally packed into symbols. Hence, throughput is calculated in symbols per second. Number of bits packed into a symbol can vary in these technologies[3]. In WiMAX, if the end to end SNR is low, only 1 bit will be transmitted per symbol. This is

known as QPSK modulation. If the end to end throughput is very high, 6 bits can be transmitted per symbol. This is known as 64-QAM modulation.

Table 1.2 – symbol rate for different modulation indices

Modulation	**Bits/symbol**
Bpsk	1
Qpsk	2
16-QAM	4
64-QAM	6

In WiMAX, if the node is closer to the base station, the end to end SNR will be high. Hence the base station will adaptively communicate at a higher modulation. Hence, the throughput will be higher for nodes with high SNR.

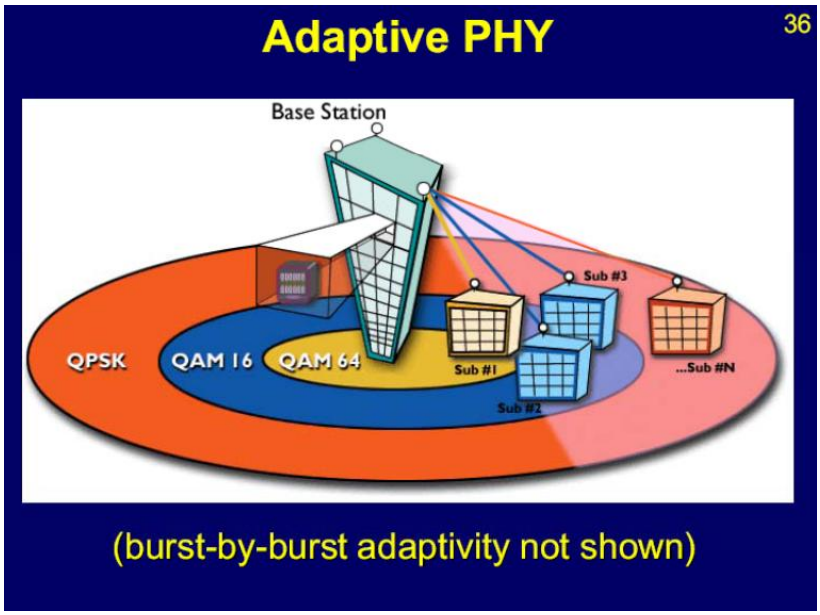

Figure 1.6 – Adaptive Modulation Coding

If we use a directional antenna, the gain will be higher, the end to end SNR will be high, the nodes will communicate with a high modulation index, the throughput will go up.

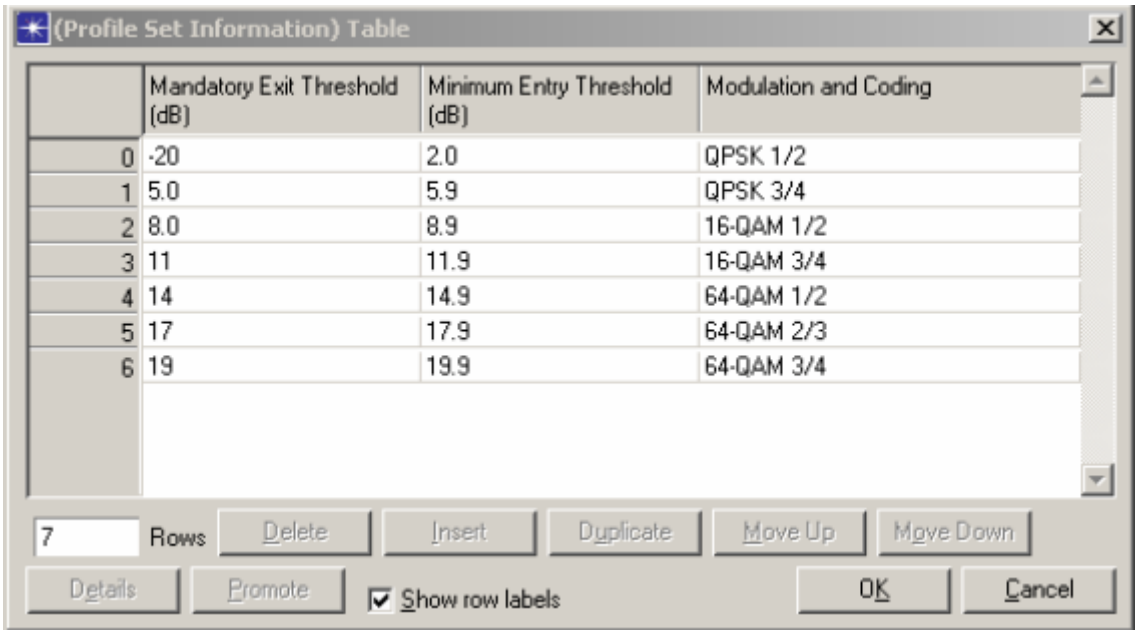

Figure 1.7 – entry and exit SNR threshold for different modulation in OPNET simulator

Project Definition

We have worked on a resource allocation problem to maximize the end to end throughput. Suppose we have a network with *m* mobile stations and *n* relay stations, we have developed an algorithm with assigns each mobile station to a relay station (relay station uses directive antennas with a beam width θ) such that the throughput of the entire network is maximized.

CHAPTER 2

ANTENNA PATTERN CREATION IN OPNET MODELER

In the real world, antenna coordinate systems are defined by Azimuth plane and the Elevation plane. In OPNET, the Azimuth plane is defined by the parameter θ and the Elevation plane is defined by the parameter Φ. In OPNET Modular, θ and Φ are incremented in steps of 5 degrees, and each and every combination of θ and Φ has to be assigned a gain value. The units of θ and Φ are degrees and the unit of gain is dB.

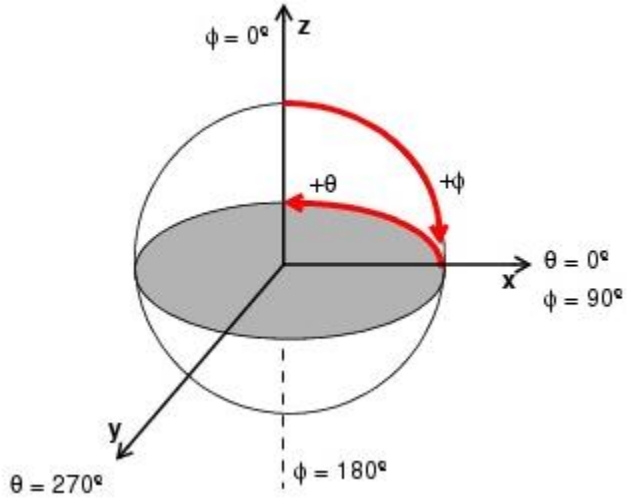

Figure 2.1 – θ (Azimuth plane) and Φ (Elevation plane) in OPNET

The range of θ is [0 360] and the range of Φ is [0 180]. As mentioned before both θ and Φ are incremented in steps of 5. Hence in OPNET θ has 72 data points and Φ has 36 data points, and each combination of θ and Φ has to be assigned a gain value. In other words, to create a valid antenna pattern file, for Φ equal to 0, we should assign 72 gain values for θ. For Φ equals 1 we should assign 72 gain values for θ and so

on up till Φ equals 36 for which we should assign 72 gain values for θ. Hence OPNET antenna patterns are defined in a 36 by 72 array[4].

```
double  dvec_2 [] =
    {
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20,
    };
```

Figure 2.2 – 36 gain definitions for a value of θ

In Figure 2.2, the integer 2 in dev_2 says that these gain values are defined for Φ between 10 degrees to 15 degrees. Like this, we will have definitions from dev_0 till dev_35.

Methods of Creating Antenna Patterns

There are 2 methods of creating antenna patterns in OPNET.

- Antenna Pattern Editor
- External Model Access (EMA)

Antenna Pattern Editor

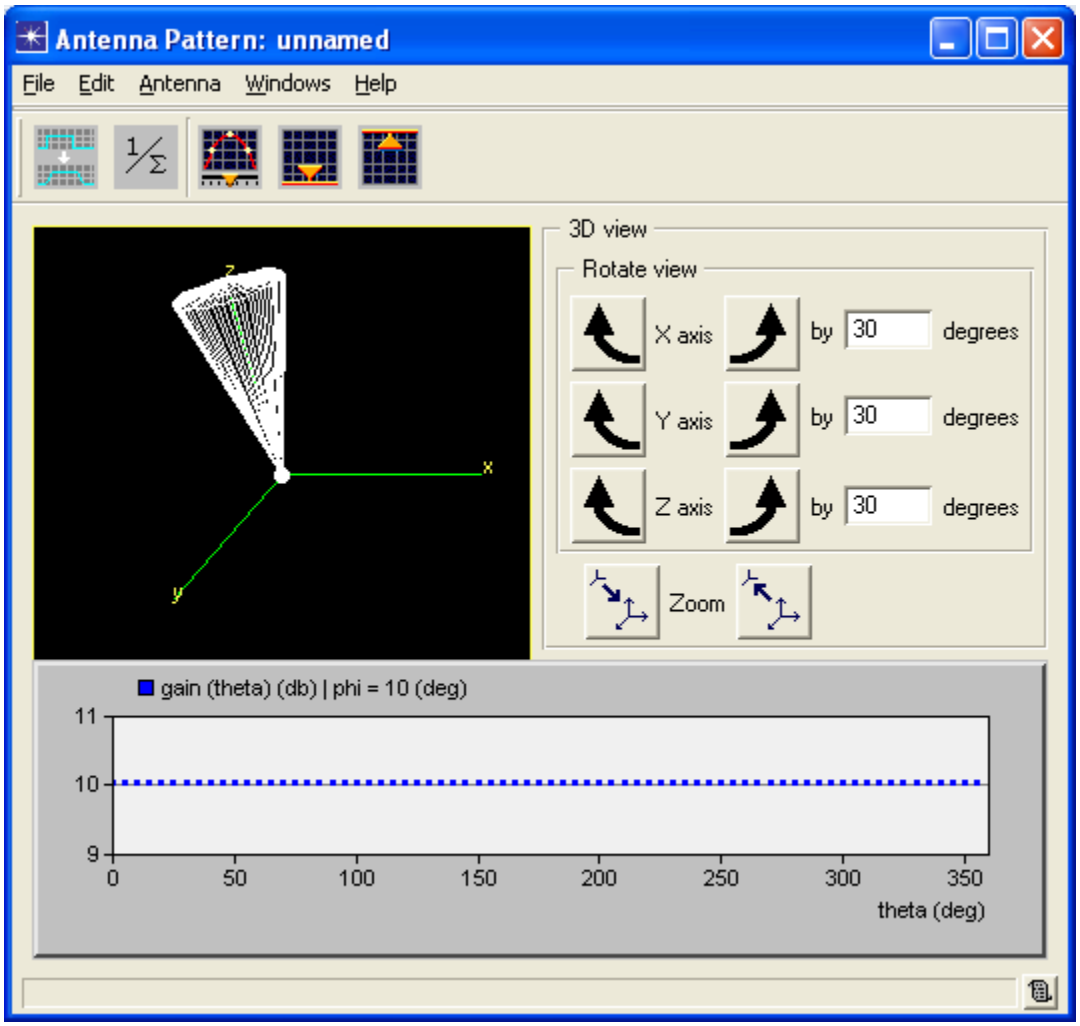

Figure 2.3 – Antenna Pattern of gain 10 dB for θ from 0 to 15 degrees

The Antenna Pattern Editor lets you create, edit, or view antenna patterns. An antenna pattern is a set of two-dimensional functions that establish the gain in dB of an antenna in three-dimensions. In an Antenna Pattern Editor, we have to click and define the gain values. There are 2 steps to create an antenna pattern using antenna pattern editor

- Select conical slices of the antenna pattern for editing.
- Use a two-dimensional graph to enter the gain pattern for the current slice.

Because the antenna pattern table is manipulated graphically within the Antenna Pattern Editor, it is inevitably somewhat imprecise.

External Model Access

EMA is a library of procedures, supplied with the OPNET release, which are useful for creating or querying models from an external program. EMA-based applications are programs which make calls to the EMA library in order to access OPNET models.

EMA library is a very useful for creating antenna patterns because we can create very accurate antenna patterns. Unlike the antenna pattern editor, there is no clicking involved and we can enter floating point values for gain. It is also a useful tool because using EMA, we have created a algorithm that automatically creates antenna patterns. Hence, in a mobile scenario, when the nodes are moving, the antenna pattern has to change as the nodes move. During the simulation, we can create new antenna patterns and assign it to the nodes. This cannot be done if we create antenna patterns with an antenna pattern editor.

Automatic Creation of Antenna Patterns

In figure 2.4, the angle between SS_1, BS and SS_2 is 27^{o} and the angle between SS_1, BS and SS_3 is 66^{o}. We created an algorithm that would create an accurate antenna pattern. The inputs to the algorithm are

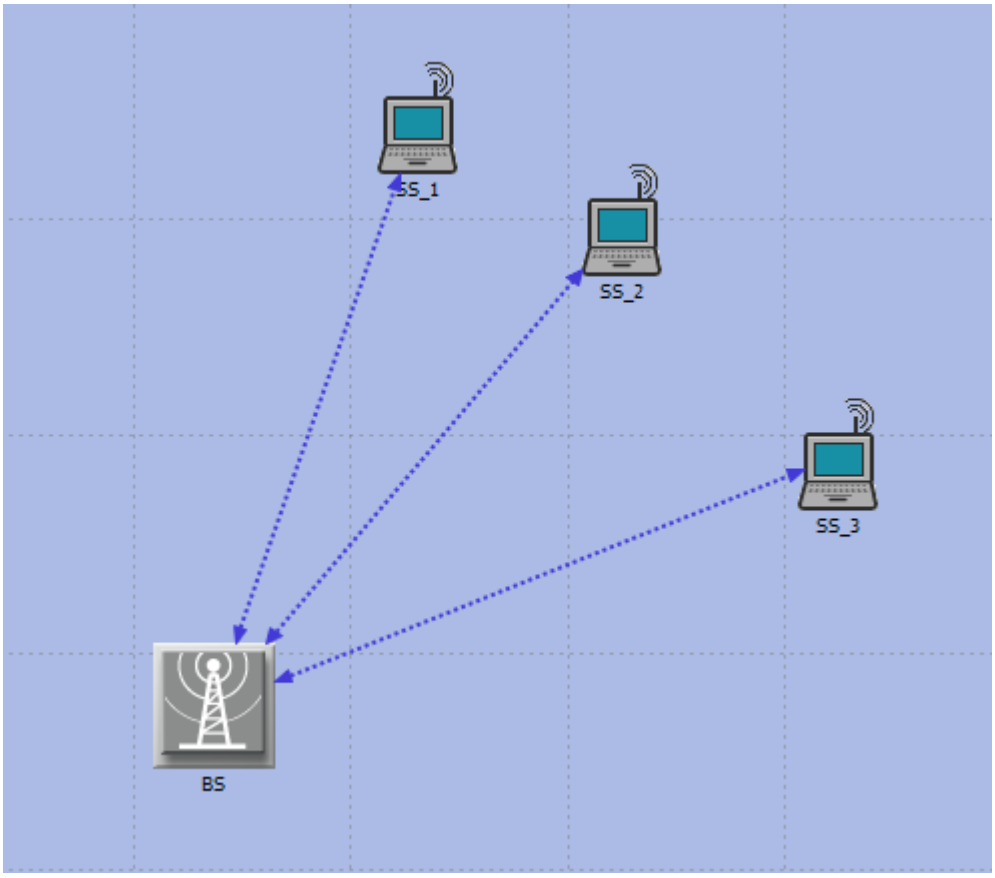

Figure 2.4 – Example network in OPNET

- Number of subscriber stations.
- The upper and lower angle between the subscriber station 1 and the remaining subscriber stations. For example the angle between SS_1 and SS_3 is 66.7^{o}, we will give the upper and lower bounds as 65 and 70. We have an upper and lower bound because as mentioned before, OPNET antenna patterns increment in steps of 5.
- Name of the antenna pattern file

The output C code is a ".em.c" file. It is a file in C language that is formatted in such a way that when it is executed, it calls the EMA library and creates an executable file.

EMA Library has a function op_mkema( ). This is a command that has to be called to create antenna patterns. op_mkema –m filename (with no extension) should be called in your command prompt. It will create an executable. The name of the executable file will be mentioned. This executable file is a ".x" format file. When this executable file is executed an antenna pattern file is created. This file is of ".pa.m" format.

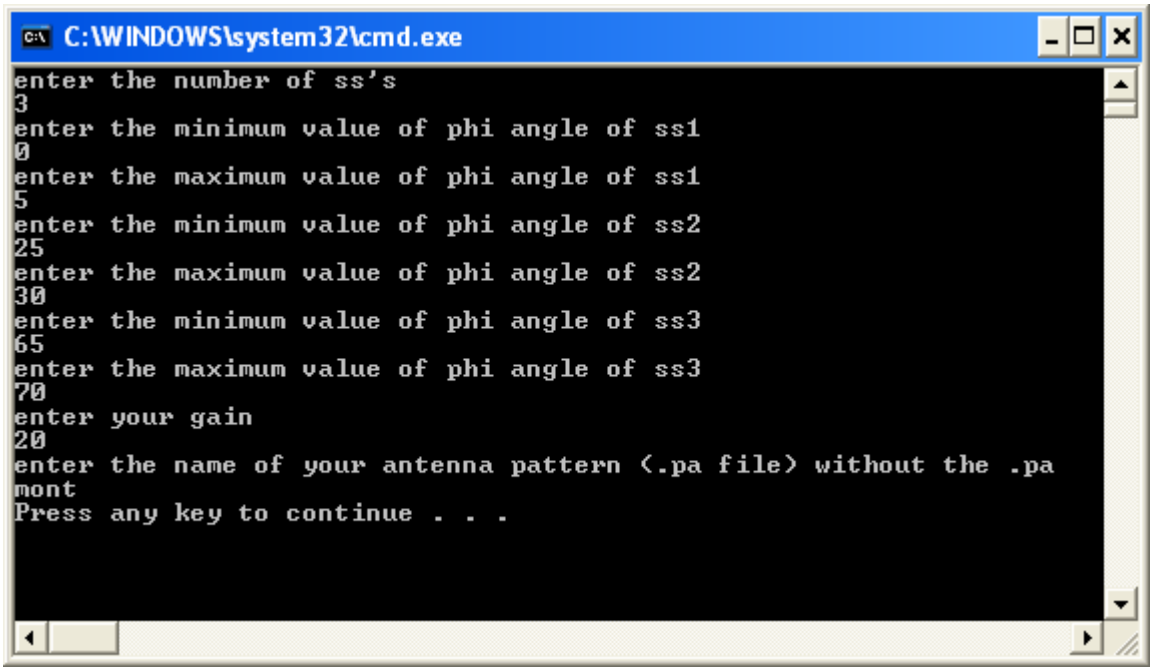

Figure 2.5 – Input to the C code

As we can see in figure 2.6, the name of the file at the output of the C code is "abcdefg.em.c". After calling the op_mkema( ) command, an executable file is created known as "abcdefg.dev32.i0.em.x". After executing this file, "mont.pa.m" antenna pattern file is created. This pattern can be seen in figure 2.7. As we can see, the angle between the first cone and the second cone is 27^{o}, and the angle between the first cone and the third cone is 66^{o}. Hence, this antenna pattern file can be used by the BS to serve the subscriber stations.

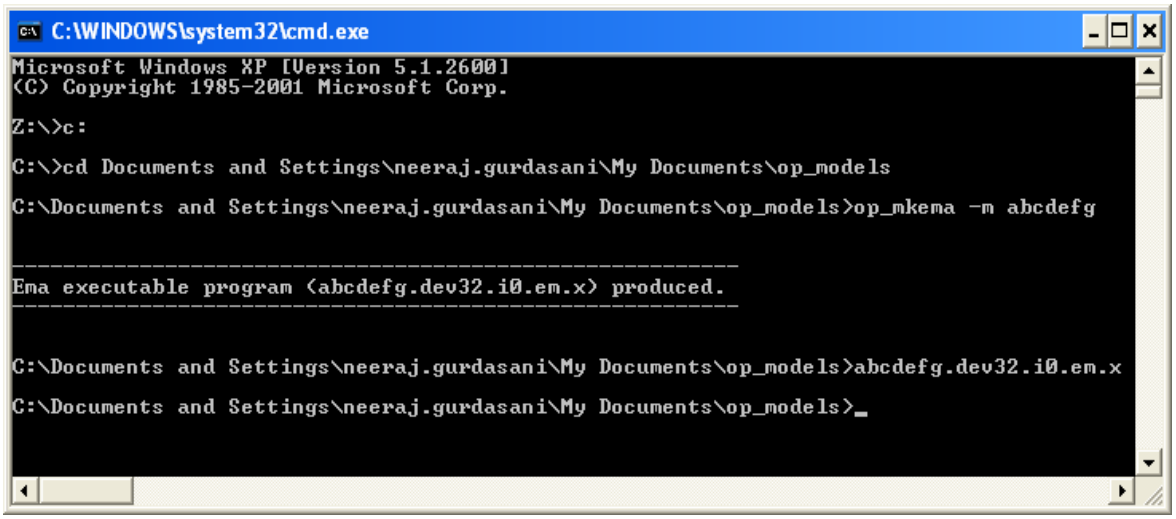

Figure 2.6 – Creation of antenna pattern

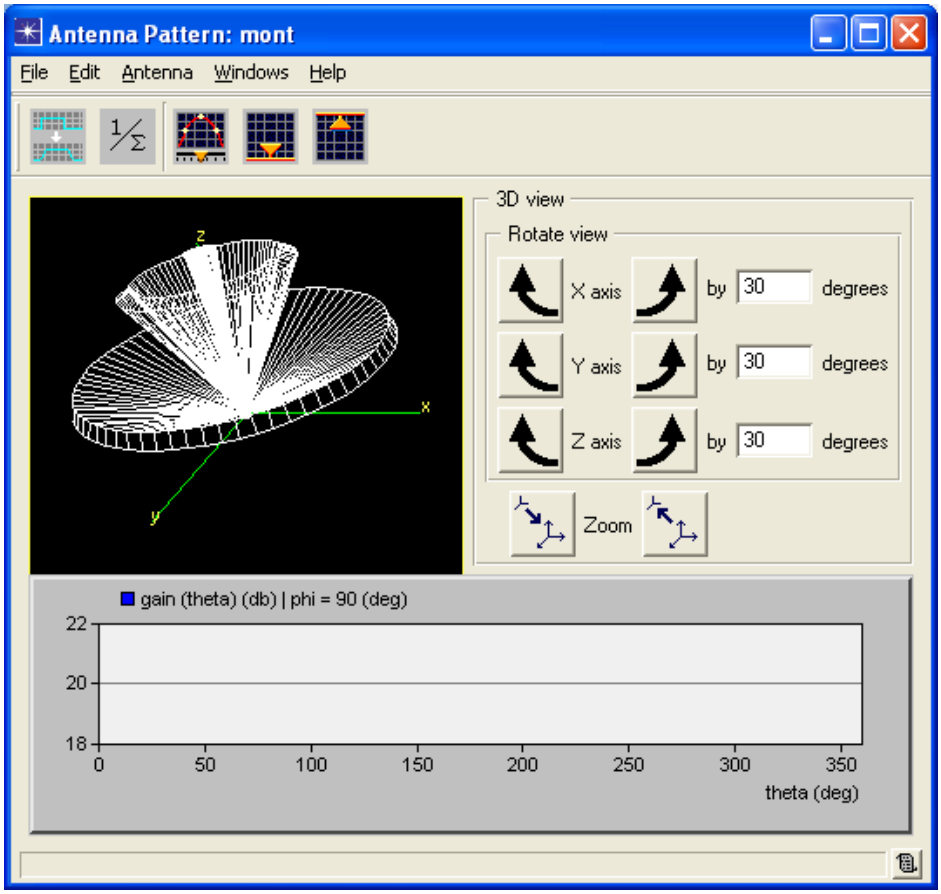

Figure 2.7 – Antenna Pattern “mont”.

Cosine Squared Gain

In the current algorithm, for a given value of Φ, θ is always constant. Like in Figure 2.7, the gain values are either 20 dB or -20dB.

```
double  dvec_7 [] =
    {
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20,
    };
```

Figure 2.8 - given value of Φ, θ is always constant

To make the antenna pattern more realistic, we made the gain follow a cosine squared curve. For example in a 90 degree sector antenna, the gain would be a maximum (max gain) for theta = 45 degrees, and then fall to -20dB for theta = 0 and theta = 90 degrees. This is done to make the physical layer calculations more realistic.

```
double  dvec_0 [] =
    {
-20.000000, -18.888889, -17.777778, -16.666667, -15.555555, -14.444444, -13.333333,
-12.222222, -11.111111, -10.000000, -8.888889, -7.777779, -6.666668, -5.555557,
-4.444447, -3.333336, -2.222225, -1.111115, -0.000004, 1.111107, 2.222218, 3.333328,
4.444439, 5.555550, 6.666660, 7.777771, 8.888882, 9.999992, 11.111103, 12.222214,
13.333324, 14.444435, 15.555546, 16.666656, 17.777767, 18.888878, 20.000000, 18.888889,
17.777778, 16.666667, 15.555555, 14.444444, 13.333333, 12.222222, 11.111111, 10.000000,
8.888889, 7.777779, 6.666668, 5.555557, 4.444447, 3.333336, 2.222225, 1.111115, 0.000004,
-1.111107, -2.222218, -3.333328, -4.444439, -5.555550, -6.666660, -7.777771, -8.888882,
-9.999992, -11.111103, -12.222214, -13.333324, -14.444435, -15.555546, -16.666656,
-17.777767, -18.888878,
    };
```

Figure 2.9 – Gain values following a cosine squared curve

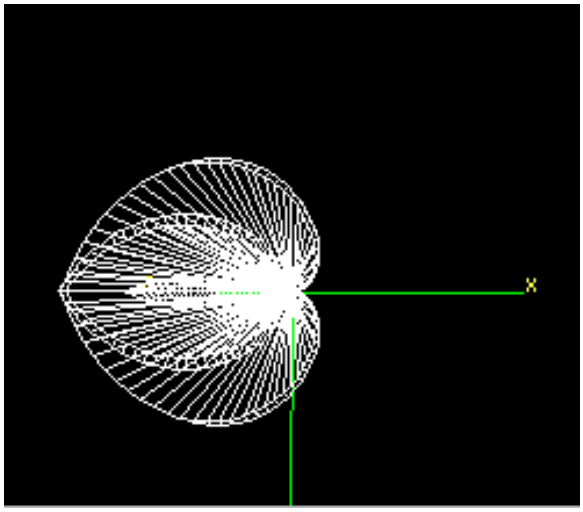

Figure 2.10 – Upper view of antenna pattern “mont” for the simulation as in figure 2.5

Implementation in OPNET

The main purpose of this algorithm was to automate the process of creating antenna patterns with the accuracy which is not there in an Antenna Pattern Editor. In OPNET Modeler, all the physical layer code is present in the “wimax_phy_support.ex.c”. In this file, we have all the physical layer code pipelined into 7 stages.

In the last stage of this pipeline, where all the link layer calculations are done (Pathloss, SNR calculations) every packet is assigned an antenna pattern and the antenna aiming parameters. The antenna aiming parameters contains 3 parameters that are latitude, longitude and altitude. All these 4 parameters are defined in a function wimax_phy_mcarrier_pk_send( ). An antenna pattern can be created by our algorithm shown above and assigned to each packet to be sent. The aiming parameters can be obtained by a set of kernel function op_ima_obj_attr_set( ) in OPNET. After obtaining the aiming parameters (latitude, longitude and altitude), these values can be set to the packet by the kernal function op_ima_obj_attr_set( ).

In a stationary case, the above algorithm has to be run just once, so that doesnot effect the simulation time. But for a mobile case, the algorithm has to be run for each and every packet sent, because the antenna pattern changes because the nodes are moving. Hence, the algorithm has to be executed every time, and the function op_mkema( ) mentioned above takes 2-4 seconds to execute. Hence the simulation time gets effected.

Tracking in OPNET

The main advantage of setting antenna patterns using EMA is that this is the only way we can use directional antennas in a mobile scenario. In a mobile case, the subscriber stations will always be in motion. If we use a directional antenna without tracking, the subscriber station might go out of coverage area. We designed a way of obtaining the current coordinates of the subscriber station and updated the aiming parameters of the antenna object in the BS for every packet sent in wimax_phy_mcarrier_pk_send( ).

We used the kernel procedure op_ima_obj_pos_get( ) to get the coordinates of the subscriber station and op_ima_obj_attr_set( ) kernel procedure to set the aiming parameters of the antenna object.

```
rx_node_id11 = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_FIX, "BS");
op_ima_obj_pos_get (rx_node_id11, &latitude, &longitude,&altitude, &x_pos, &y_pos, &z_pos);
BS_lat=latitude;
BS_log=longitude;
BS_alt=altitude;
```

Figure 2.11 - op_ima_obj_pos_get( )

After getting the coordinates of the subscriber station, we had to set the aiming parameters of the antenna object in the Base Station. That code can be seen in code

attached. This has to be done for each and every packet sent to make sure the subscriber station never goes out of the antenna coverage area.

Figure 2.12 shows the scenario layout that exhibits tracking. Table 2.1 shows the scenario parameters. Figure 2.13 shows the received traffic (bits/second) when tracking code is disabled. Figure 2.14 shows the received traffic (bits/second) when tracking is enabled.

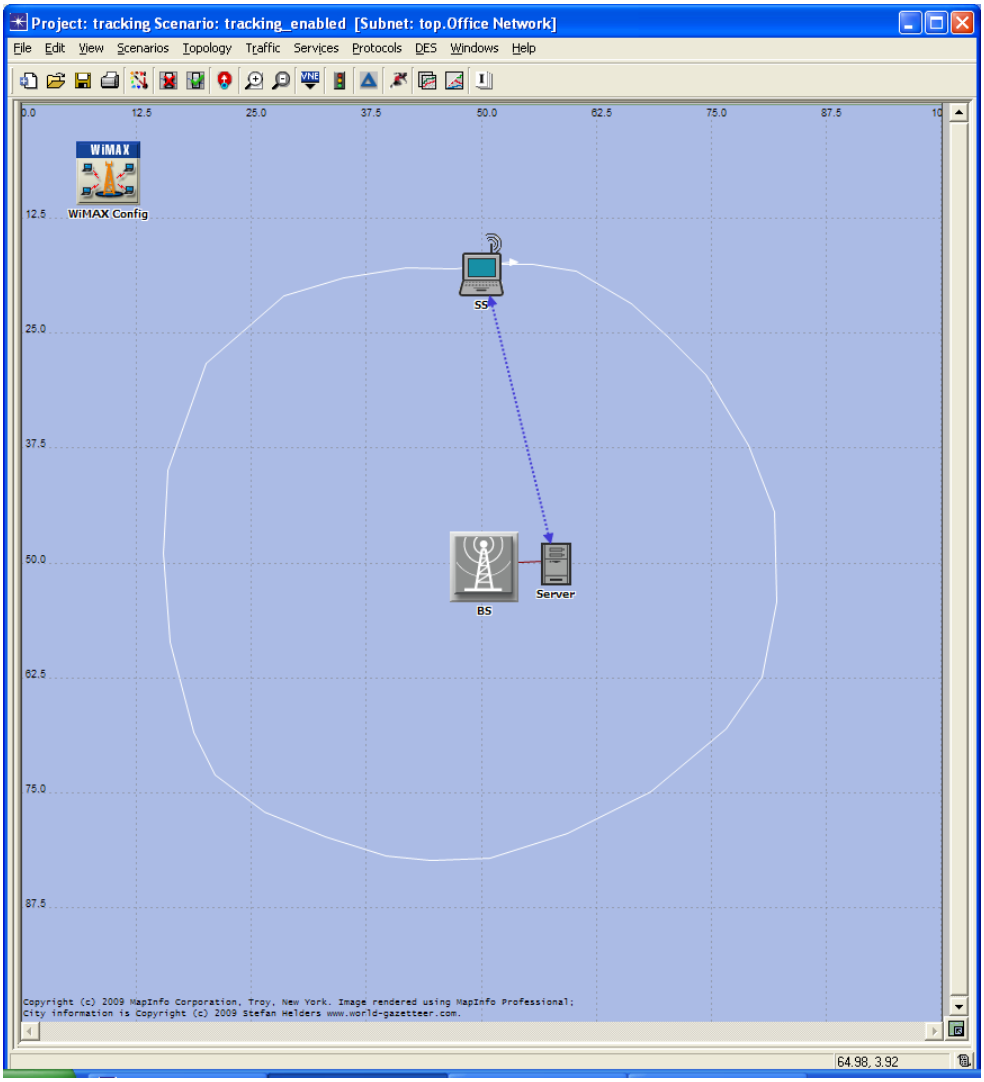

Figure 2.12 – Tracking Scenario

Table2.1 – Simulation parameters

Antenna pattern of BS	5^{o} cone antenna of gain 20dB
Antenna pattern of SS	-18dB omni-directional
BS Mobility	False
SS Mobility	True(circular trajectory)
Traffic Demand	1 Mbps

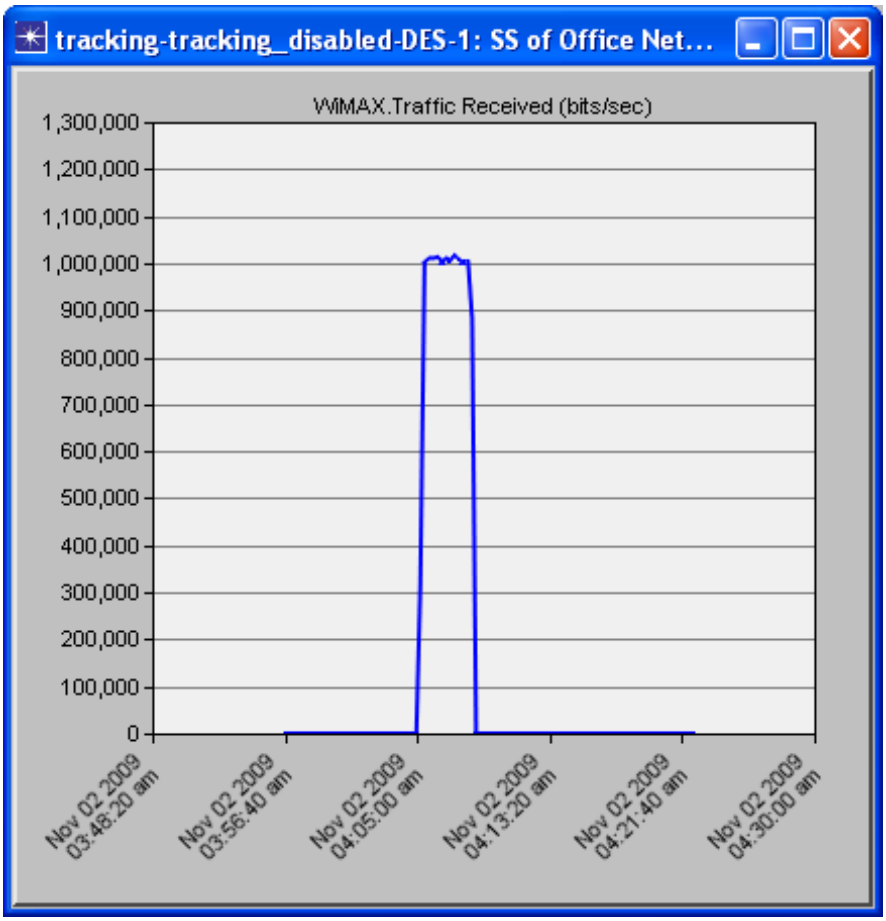

Figure 2.13 – Traffic received by SS when tracking disabled

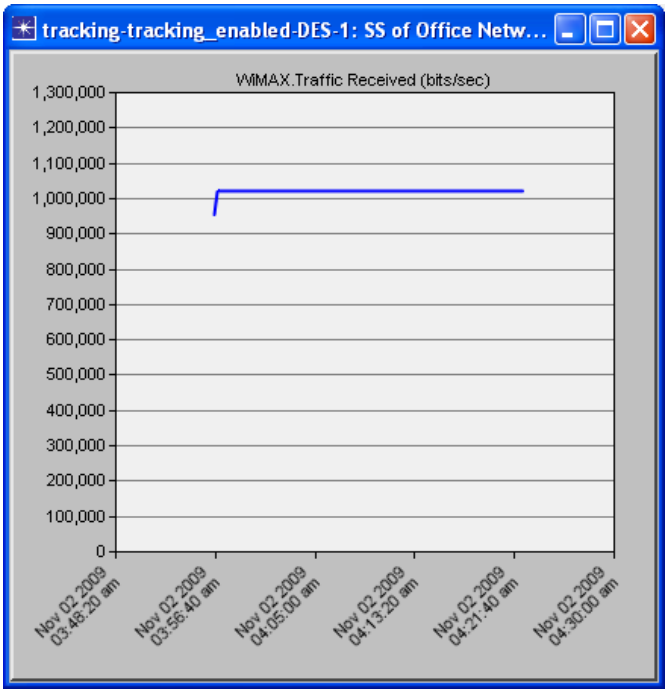

Figure 2.14 – Traffic received by SS when tracking enabled

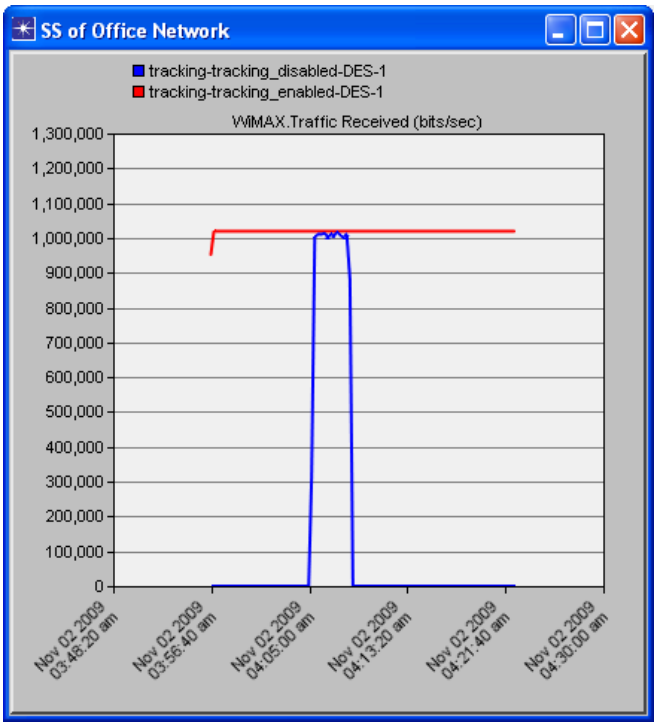

Figure 2.15 –Tracking enabled Vs Tracking disabled

As we can see in figure 2.13, when tracking is disabled, the node in motion comes in the coverage area and leaves the coverage area. That is the only time when the node SS is able to communicate with the base station. In figure 2.14, we are able to consistently receive traffic from the base station when tracking is enabled.

Pencil Beam Antenna Patterns

As we can see in figure-2.7, we have managed to create conical antenna patterns. These are directive antenna patterns where we have 72 high gain values of θ for values of Φ which come under the main lobe and 72 low gain values (-20dBi) of θ for values of Φ which don't come under the main lobe. So we are entering 72 high gain values of θ for values of Φ which come under the main lobe. But we don't have to give all the 72 data sets high gain values. If we give 3 data sets high gain values that should be enough to achieve communication. If we manage to do that, we will be creating extremely directive antenna patterns known as pencil beam antenna patterns.

```
double  dvec_7 [] =
    {
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20,
    };
```

Figure 2.16 – High gain values of θ for a conical antenna pattern

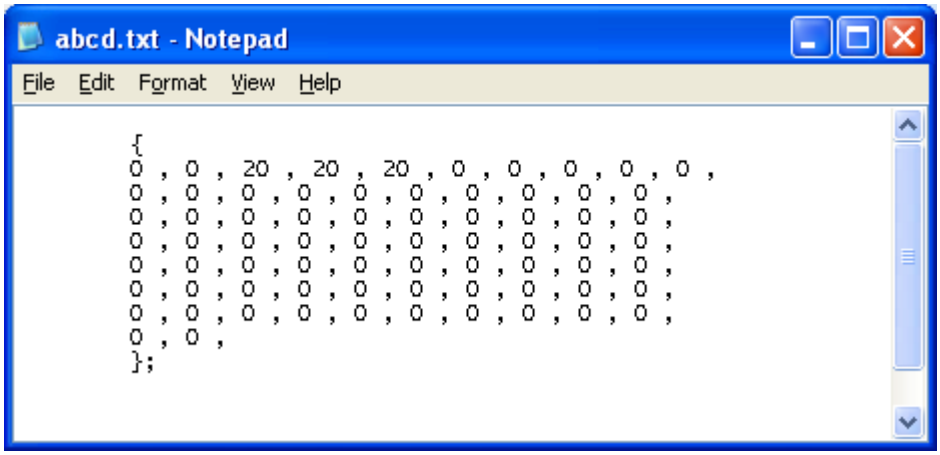

Figure 2.17 – High gain values of θ for a pencil beam antenna pattern

As we can see in figure 2.17, we just have 3 high gain values of θ for Φ that comes under the main lobe. The problem in creating this antenna pattern is which of the 72 data sets values have to be high so that communication. For that we have to do θ calculations. We have created an algorithm to solve this problem. The steps of the algorithm are

- Let the 1st Subscriber stations coordinates be (x1,y1,z1). Let the coordinates of the Subscriber stations from whom we are trying to find θ be (x2,y2,z2). By these 2 coordinates, find the equation of a sphere with (x1,y1,z1) being the center.
- Calculate the equation of a plane with the coordinates (x1,y1,z1), (x1,y1,0) and (xb,yb,zb), where (xb,yb,zb) are the coordinates of the Base Station.
- Calculate the points of intersection of the sphere and a plane. The set of satisfied values will form a circle. Select the highest point of that circle, let the coordinates of that point be (xt,yt,zt).
- Calculate the angle between (xt,yt,zt), (xb,yb,zb) and (x2,y2,z2). That is the value of θ that had to be of high gain.[5]

These steps have to be repeated for all the subscriber stations in the network (apart from the first subscriber station). The coordinates of those subscriber stations should be placed in place of (x2,y2,z2). We ran our algorithm for scenario in figure-2.4. The output pencil beam antenna pattern can be seen in figure-2.18.

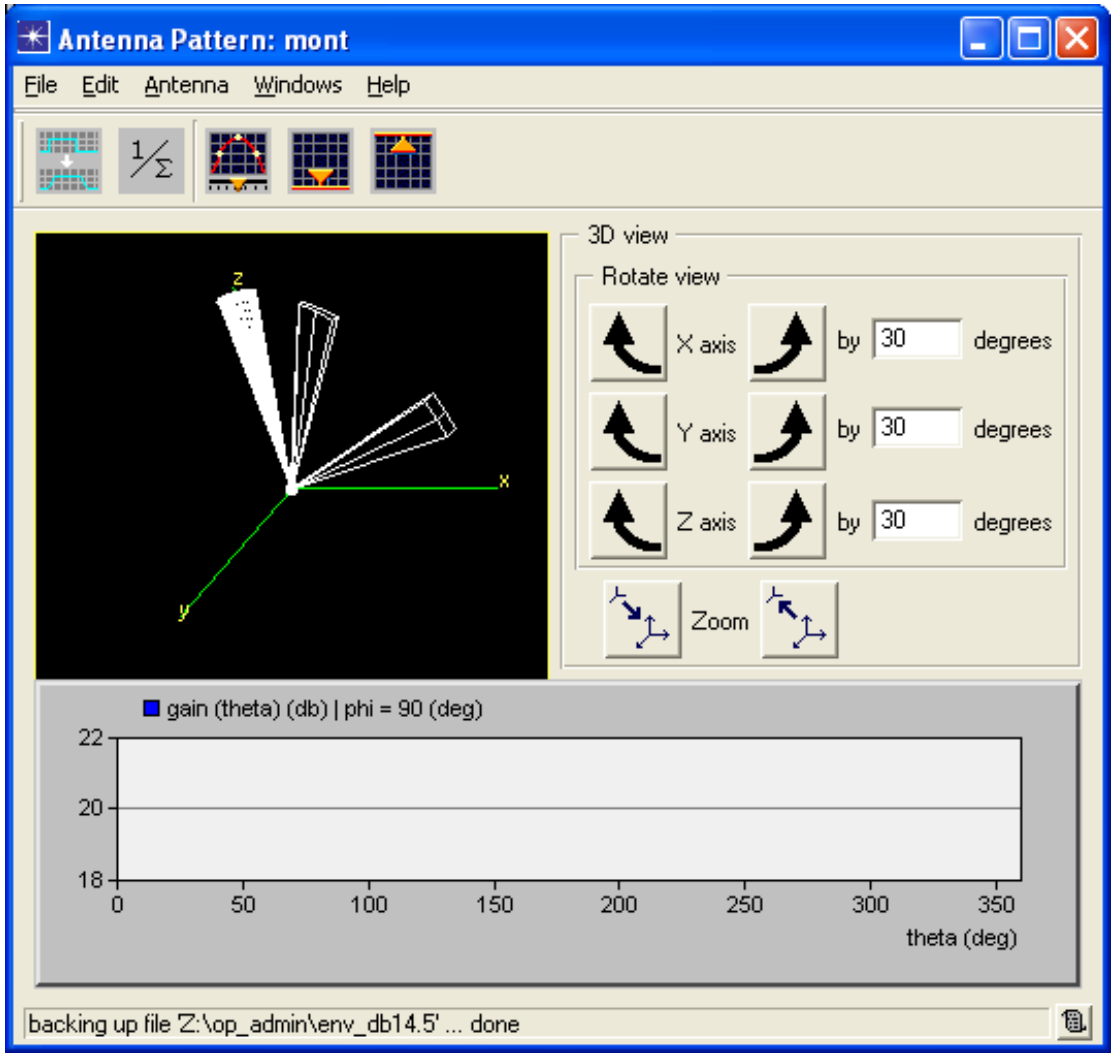

Figure 2.18 – pencil beam antenna pattern for scenario in figure 2.14

CHAPTER 3

IMPLEMENTATION OF A RELAY BEAM STEERING ALGORITHM

This chapter describes the implementation and testing of a recently invented algorithm[6] for choosing relay beam directions and assigning mobile stations to relays in such a way as to maximized the total throughput achieved in the network.

Let us imagine a network with *n* relay stations and *m* mobile stations. The problem we face is that which mobile stations should be served by which relay stations. We have worked on an optimization problem where we assign every subscriber station in the network to one of the available relay stations in the network such that the throughput of the entire network is maximized.

We created an Integer linear program (ILP) formulation to solve this NP-Complete problem. The objective function of this ILP is to maximize the data rate. To efficiently solve this problem we relax the ILP to a Linear Program (LP) by allowing real values (non-integer) for the ILP. Hence, the output of the LP may not be an integer solution. To get an absolute integer assignment, we have created a rounding algorithm that takes the output of the LP as an input and gives an integer assignment. The rounding algorithm provides the first provable approximation.

Hence to solve this problem, we have to follow these steps

- Create a ILP formulation
- Relax the ILP to an LP (run the ILP in LPSolve software)
- Run the rounding algorithm whose input the output of LP.

ILP Formulation

Let there be *m* mobile stations M_1, M_2, …M_m and *n* relay stations R_1, R_2, R_3…R_n at a given position in a plane. Let us assume that each relay station Rj has a fixed beam width angle θ(in our simulations, we have set that value to 40^o). Now the objective function of out ILP is to maximize the data rate. Let r_{ij} be the data rate achievable if M_i is within the beam of R_j. Let d be the distance between M_i and R_j. Then

$$r_{ij} = \frac{1}{(d^2 * \theta)}$$

Beam Sets are the distinct set of mobile stations that are reachable by a relay station R_j as its main lobe beam spans 360^o. Hence, each relay station spans its antenna main lobe beam and calculates all its beam sets B_{j1}, B_{j2}, …., B_{jp}.

In example network shown in figure 3.1 the beam sets for relay station RS will be

- B_{01} = 0,1
- B_{02} = 1
- B_{03} = 2
- B_{04} = 2,3
- B_{05} = 3
- B_{06} = 4
- B_{07} = 4,5
- B_{08} = 5
- B_{09} = 6
- B_{010} = 6,7

- $B_{011} = 7$

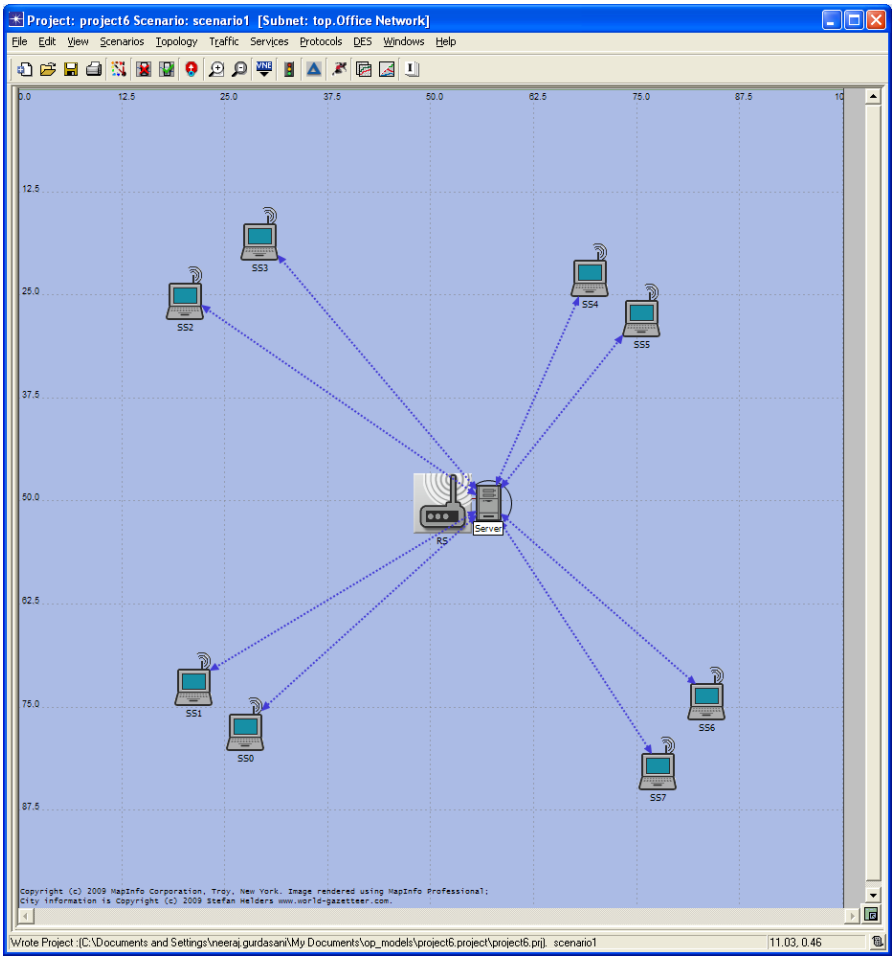

Figure 3.1 – example network

So we can say that it is a cluster of all the mobile stations that can be served when the antenna beam is aimed in the certain orientation. So the collection of these different clusters when the antenna beam spans 360^0 are called as a beam sets.

For each mobile station M_i, let S_{ij} be the collection of beam sets for R_j that contain M_i i.e $M_i \in B_{jk} <=> k \in S_{ij}$.

Variables and Constraints

Now for a mobile station M_i and a relay station R_j, the variables of the ILP are

- x_{ij} (x_{ij} = 1 implies mobile station M_i uses relay station R_j)
- S_{jk} (S_{jk} = 1 implies relay station R_j uses beamset B_{jk})

The constraints of the ILP are

1. $0 \leq x_{ij} \leq 1$
2. $0 \leq s_{jk} \leq 1$
3. $\forall i : \sum_{j=1}^{n} x_{ij} \leq 1$
4. $\forall j : \sum_{k=1}^{n_j} s_{jk} = 1$
5. $\forall j : \sum_{k \in s_{ij}} s_{jk} \geq x_{ij}$

The objective function of this ILP is $\sum_{ij} r_{ij} x_{ij}$

The 3rd constraint of the ILP makes sure that a mobile station selects at most one relay station. The 4th constraint of the ILP makes sure that a relay station selects exactly one beamset. The 5th constraint of the ILP makes sure that if Mi chooses R_j, then R_j must select a beamset that contains M_i.

We encoded the whole ILP in the C language. The input to this code is going to be coordinates of the nodes. The output of this code going to be an ILP formulation. Figure 3.2 shows the ILP formulation for the network in figure 3.1. Figure 3.2 shows the

output of the LPSolver software when the input is the text shown in figure-3.1. In figure 3.2, we can see that x_{40} and x_{50} are 1, which implies the maximum throughput with a 40^{o} antenna in an RS can be achieved when RS serves mobile station 4 and 5.

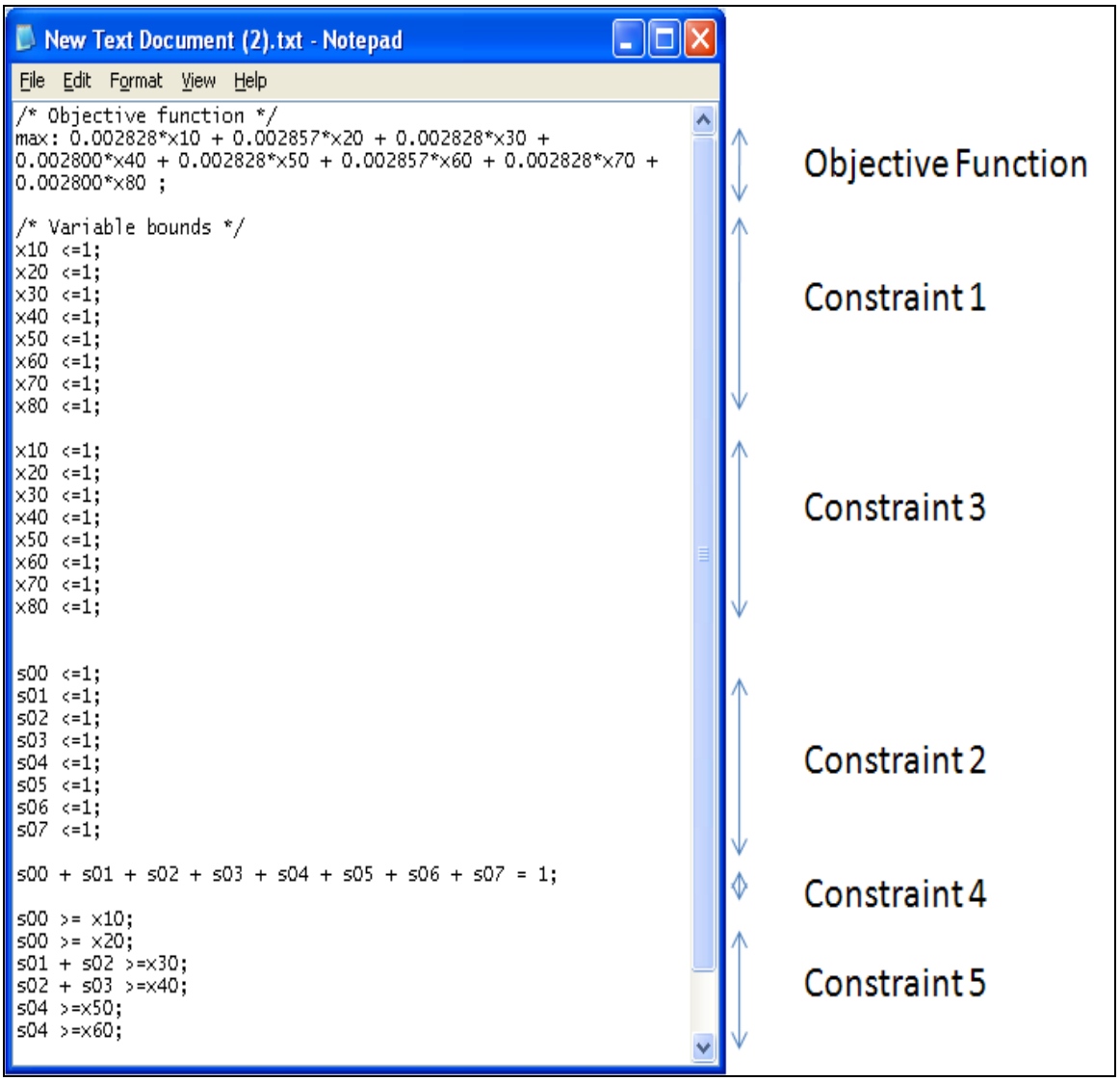

Figure 3.2 – The ILP created

Variables	result
	0.005685
x00	0
x10	0
x20	0
x30	0
x40	1
x50	1
x60	0
x70	0
x01	0
x02	0
x03	0
x04	0
x05	0
x06	0

Figure 3.3 – output of LP Solve

We relax the ILP to an LP. The issue with this is that the output of the LP may not be an integer i.e. we may have a case that if we have more than one relay stations, then the output of the LP may say the a mobile station is half served by one relay and half served by other relay ($x_{40} = 0.5$ and $x_{41} = 0.5$). The same can be the case for s_{jk}'s . The LP may say $s_{02} = 0.5$ and $s_{03} = 0.5$ which means relay station 0 should half serve mobile station 2 and half serve mobile station 3. Even though the cases mentioned above are possible, they would be impractical, because these outputs mean that the relay station has break a connection, steer its antenna, remake a connection, and then go back. This process would take some time, hence the throughput of the system will be affected beating the whole point of this algorithm. So, it would be best if the output of our algorithm is an integer solution.

Rounding Algorithm

We have created a Rounding Algorithm that would take the output of the LP as its input and give the best integer solution. The following are the steps of the rounding algorithm

1. Calculate the reward y_{jk} of each beam set B_{jk}, where
   i. $y_{jk} = \sum_{i \in B_{jk}} r_{ij} x_{ij}$
2. Choose the beam set B_{jk_j} with the highest reward y_{jk_j}.
3. For each mobile station M_i, let $A_i = \left\{ j \middle| i \in B_{jk_j} \right\}$ be the set of available relay station after the beam directions have been chosen at the end of step 2. If A_i has more than one relay station, choose the relay station with the maximum data rate.

Steps 1 and 2 are calculating the data rates achieved by different beam sets of a relay station. The beam set which has the maximum data rate is chosen i.e we are fixing the orientation of the antenna beam of a relay station. This is done for each and every relay station. In Steps 3 for each mobile station, we check how many relay station serve that mobile station. If we have more than one relay station, then we choose the relay station that provides it the maximum data rate. The rounding Algorithm is a $\frac{1}{2\pi/\theta_{min}}$ – approximation algorithm in polynomial time[6]. MAX-MSRS-THROUGHPUT ALGORITHM is proven to be NP-Complete.

Simulation using OPNET Coordinates

We run our ILP formulation and our rounding algorithm on the example network shown in figure-3.4.

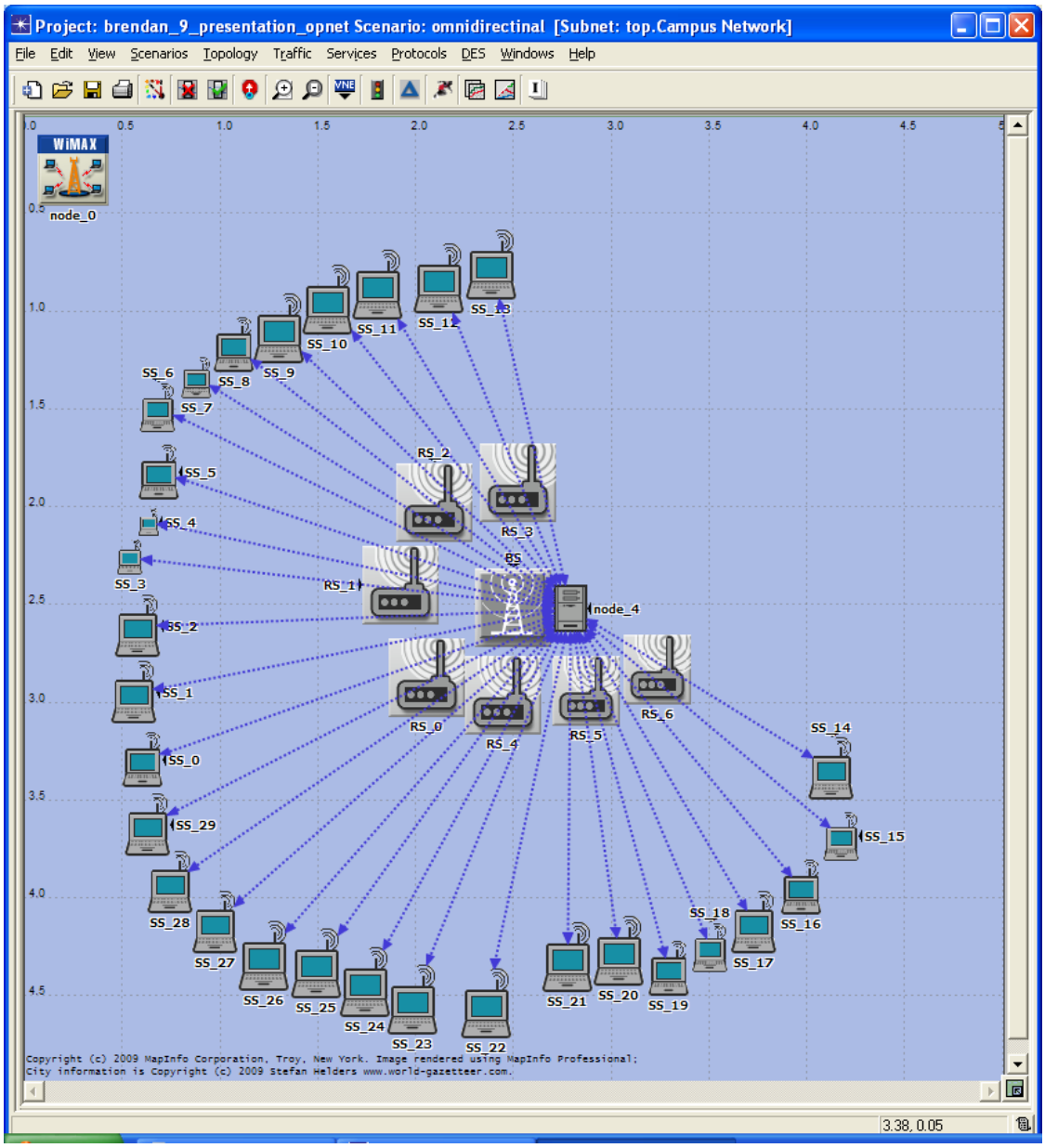

Figure 3.4 – Example Network

x00	1	x01	0	x02	0	x03	0	x04	0	x05	0	x06	0
x10	1	x11	0	x12	0	x13	0	x14	0	x15	0	x16	0
x20	1	x21	0	x22	0	x23	0	x24	0	x25	0	x26	0
x30	0	x31	1	x32	0	x33	0	x34	0	x35	0	x36	0
x40	0	x41	1	x42	0	x43	0	x44	0	x45	0	x46	0
x50	0	x51	1	x52	0	x53	0	x54	0	x55	0	x56	0
x60	0	x61	1	x62	0	x63	0	x64	0	x65	0	x66	0
x70	0	x71	1	x72	0	x73	0	x74	0	x75	0	x76	0
x80	0	x81	0	x82	1	x83	0	x84	0	x85	0	x86	0
x90	0	x91	0	x92	1	x93	0	x94	0	x95	0	x96	0
x100	0	x101	0	x102	1	x103	0	x104	0	x105	0	x106	0
x110	0	x111	0	x112	1	x113	0	x114	0	x115	0	x116	0
x120	0	x121	0	x122	0	x123	1	x124	0	x125	0	x126	0
x130	0	x131	0	x132	0	x133	1	x134	0	x135	0	x136	0
x140	0	x141	0	x142	0	x143	0	x144	0	x145	0	x146	1
x150	0	x151	0	x152	0	x153	0	x154	0	x155	0	x156	1
x160	0	x161	0	x162	0	x163	0	x164	0	x165	0	x166	1
x170	0	x171	0	x172	0	x173	0	x174	0	x175	0	x176	1
x180	0	x181	0	x182	0	x183	0	x184	0	x185	1	x186	0
x190	0	x191	0	x192	0	x193	0	x194	0	x195	1	x196	0
x200	0	x201	0	x202	0	x203	0	x204	0	x205	1	x206	0
x210	0	x211	0	x212	0	x213	0	x214	0	x215	1	x216	0
x220	0	x221	0	x222	0	x223	0	x224	1	x225	0	x226	0
x230	0	x231	0	x232	0	x233	0	x234	1	x235	0	x236	0
x240	0	x241	0	x242	0	x243	0	x244	1	x245	0	x246	0
x250	0	x251	0	x252	0	x253	0	x254	1	x255	0	x256	0
x260	0	x261	0	x262	0	x263	0	x264	1	x265	0	x266	0
x270	1	x271	0	x272	0	x273	0	x274	0	x275	0	x276	0
x280	1	x281	0	x282	0	x283	0	x284	0	x285	0	x286	0
x290	1	x291	0	x292	0	x293	0	x294	0	x295	0	x296	0

Figure 3.5 – LP- Output

The example network shown in figure-3.4, we have 1 Base-Station, 1 Ethernet Server, 6 relay Stations and 30 Sub Scriber Stations. Each subscriber station is 750,000 bits/second traffic from the Ethernet server. Hence, all the traffic is downlink. The traffic goes from the server, to the base station, to the relay stations to the subscriber stations.

We run our algorithm on the example network in figure-3.4. We take the coordinates of the nodes from OPNET and give it as an input to the algorithm. The output of the algorithm is an ILP formulation as shown in figure-3.2. The ILP formulation is given to the LPSolver software. The output of the LPSolver in shown in figure-3.5. This is given as an input to the rounding algorithm. The output of the rounding is shown in figure-3.6. Figure 3.7 shows the graphical interpretation of figure-3.6.

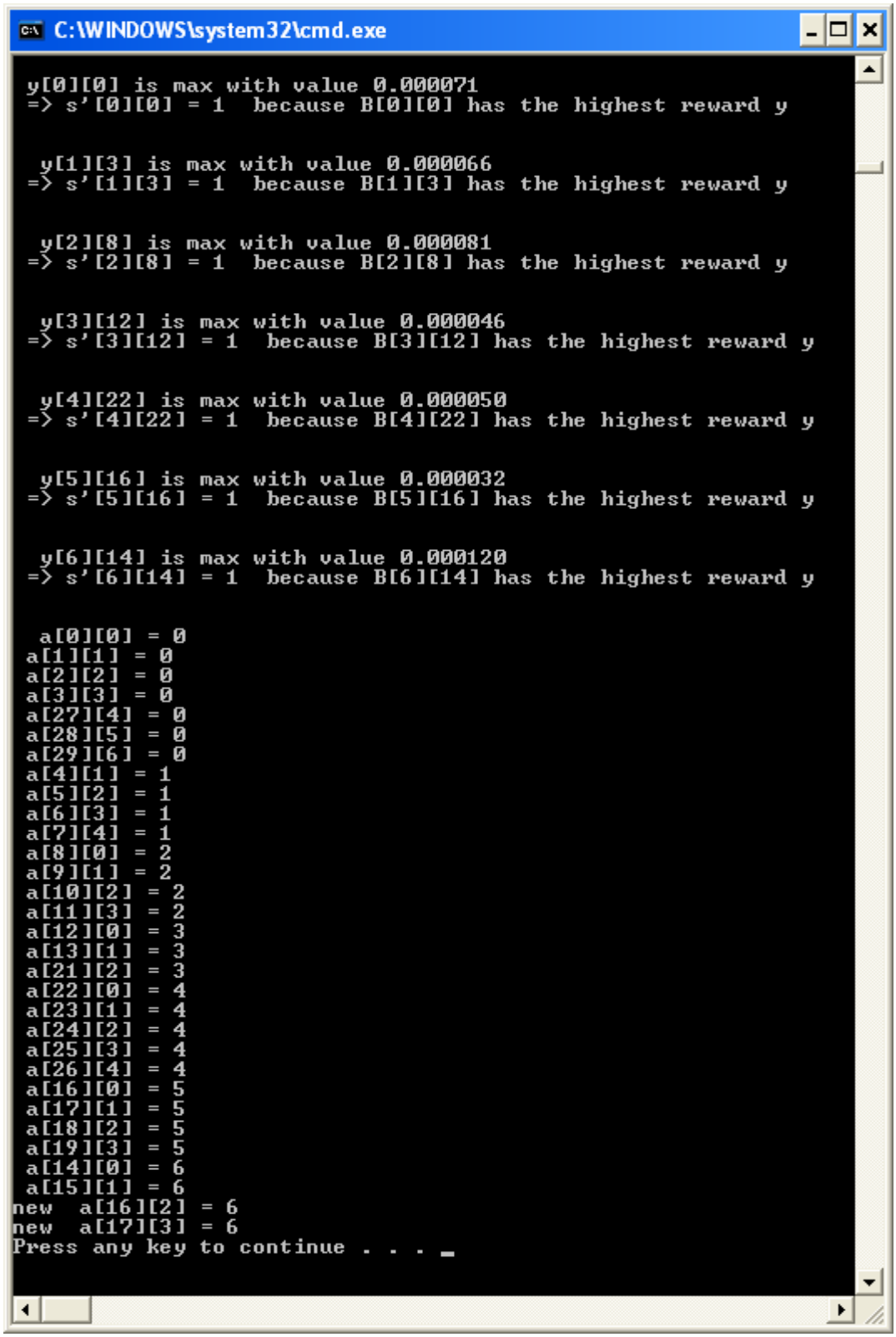

Figure 3.6 –Output of rounding algorithm

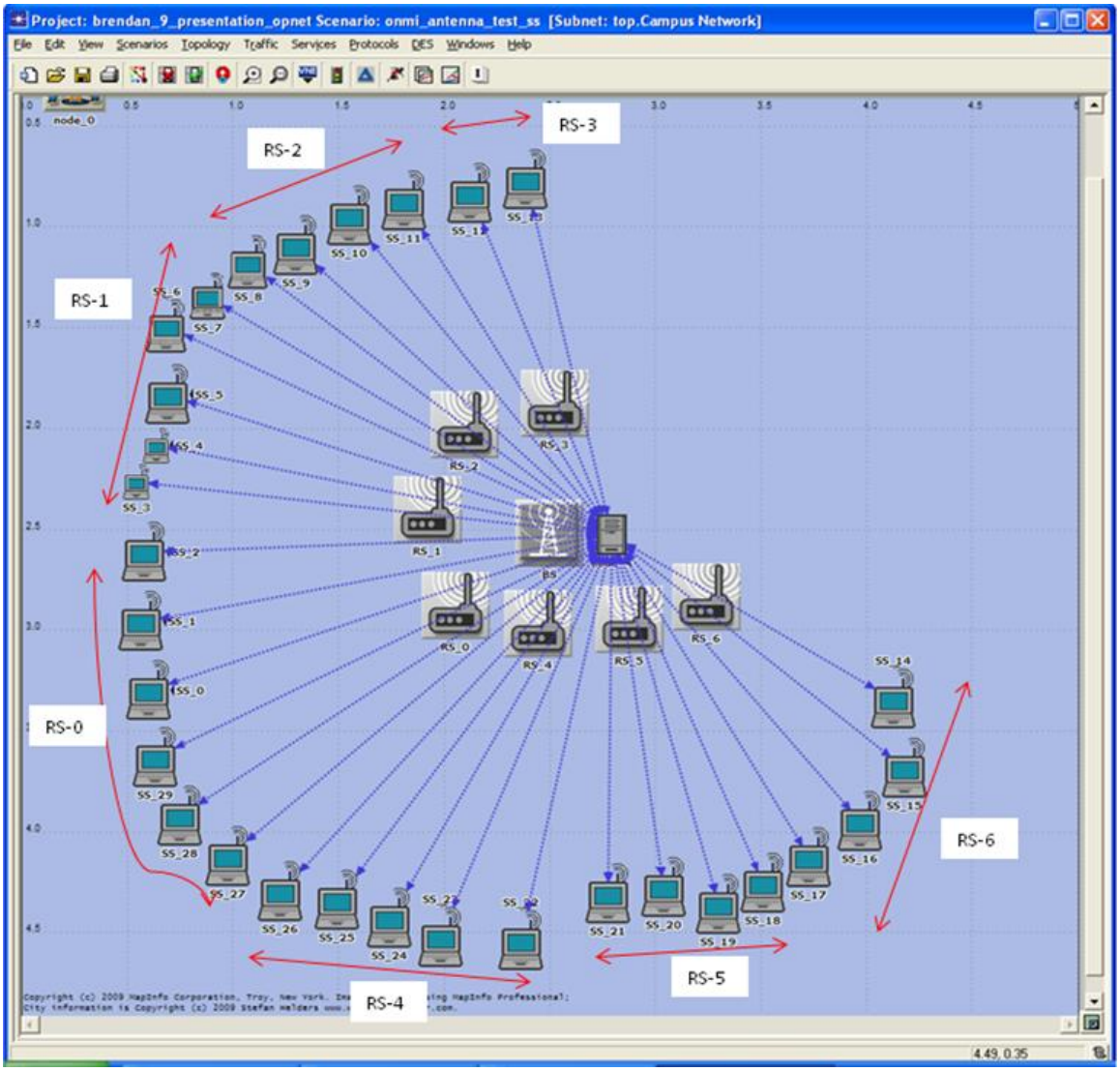

Figure 3.7 - Assignment

In figure-3.5, $x_{10} = 1$ implies sub scriber station 1 is served by relay station 0. In figure-3.6 if a[22][0] = 4, means that subscriber station 22 will be served by relay station 4. In that same figure, y[1][3] is the maximum value means relay station 1 should should use its 3rd beamset because that beamset has the maximum datarate. We can see from figure-3.7, node 22 is served by relay station 4.

Simulation on Random coordinates

We run simulations on networks of different sizes where the relay stations and the subscriber stations were randomly places using the rand function in C. The number of relay station randomly varied from 1 to 10 and the number of subscriber stations randomly varied from 10 to 100. In each simulation we compared the throughput provided by the linear program (LP), throughput provided by the Integer Linear program (ILP) and throughput provided by the rounding algorithm.

Table3.1 – Simulation results for different Network Sizes

Scenario Size	LP Throughput	ILP Throughput	Rounding Throughput	% of Optimality
RS=7 SS =22 Random Instance	1.084	1.079	0.876	81.1%
RS=4 SS =51 Random Instance	1.804	1.784	1.583	88.7%
RS=2 SS =42 Random Instance	0.925	0.925	0.589	63.7%
RS=4 SS =48 Random Instance	1.762	1.729	1.400	80.9%
RS=10 SS=10 Random Instance	6.796	6.791	6.249	92.0%
RS=8 SS=40 NP-Hard Instance	40	40	30	75.0%
RS=8 SS=44 NP-Hard Instance	44	44	28	63.63%
RS=8 SS=52 NP-Hard Instance	52	52	34	65.3%

As we can see from table 3.1, the performance of the rounding algorithm increases as the network size increases.

CHAPTER 4

SIMULATION REULTS

For the assignment shown in figure 3.7, we run the simulation using directional antennas and omni-directional antennas and compared their respective throughputs. Table 4.1 shows the simulation setting

Table 4.1 – Simulation parameters

Number of BS	1
Number of Servers	1
Number of RS	7
Number of SS	30
Traffic Demand for each SS	733,200 bits/sec
Which relay serves which SS	Figure 3.7
Area of network	5km x 5km
Simulation 1	Omni-directional antenna
Simulation 2	directional antenna
Simulation 1 BS and RS Gain	5 dBi
Simulation 2 BS and RS Gain	15 dBi
Simulation 1 and Simulation 2 SS gain	-1 dBi

The blue line in figure 4.1 is traffic sent by BS using directional antennas while the red line is traffic sent by BS using omni-directional antennas. Figure 4.2 is the sum of traffic sent by BS of relay using omni-directional antennas. Figure 4.3 is the sum of traffic sent by BS of relay using directional antennas. Figure 4.4 is the sum of traffic received by SS in simulation 1(omni-directional antennas). Figure 4.5 is the sum of traffic received by SS in simulation 2(directional antennas). Table 4.2 compares the results between simulation 1 and simulation 2.

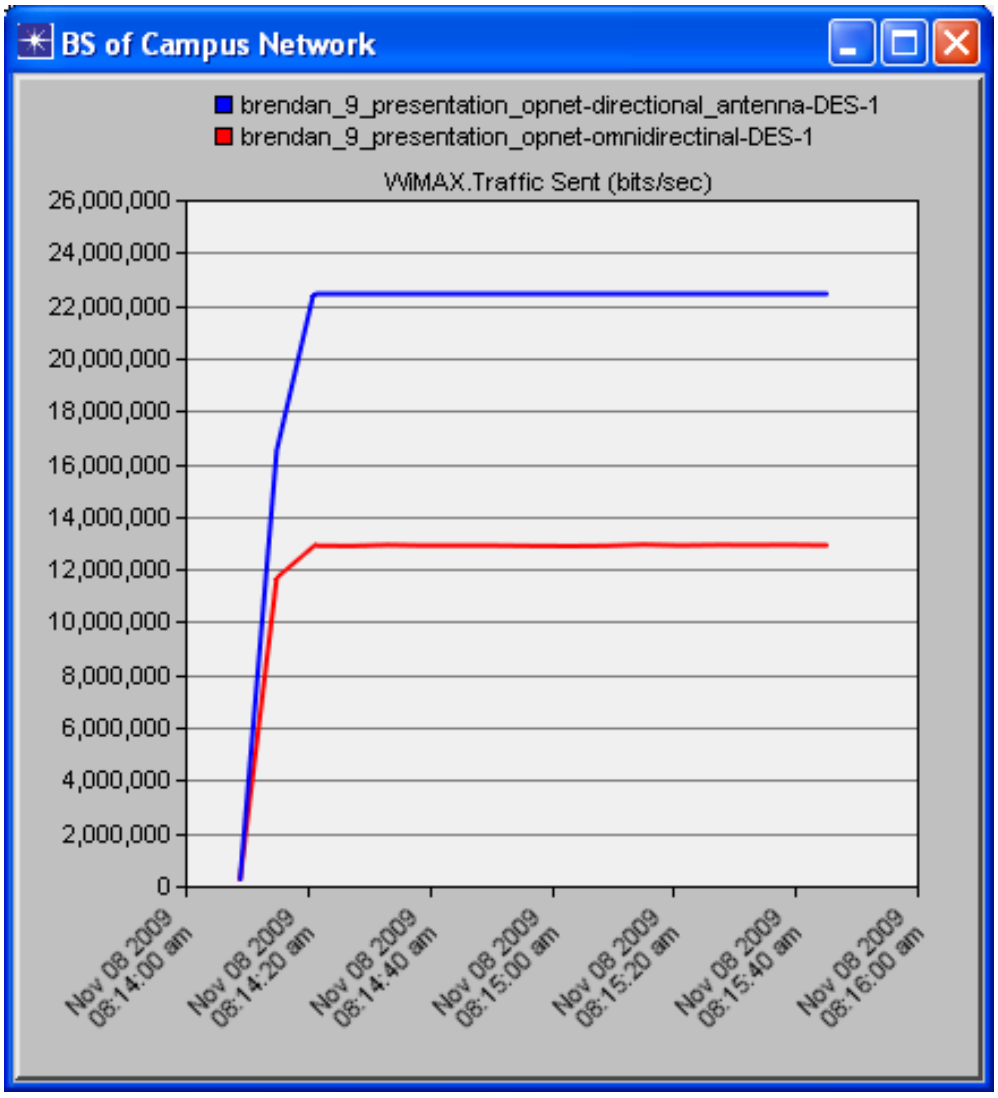

Figure 4.1 – traffic sent by BS of omnidirectional (red curve) VS directional(blue curve)

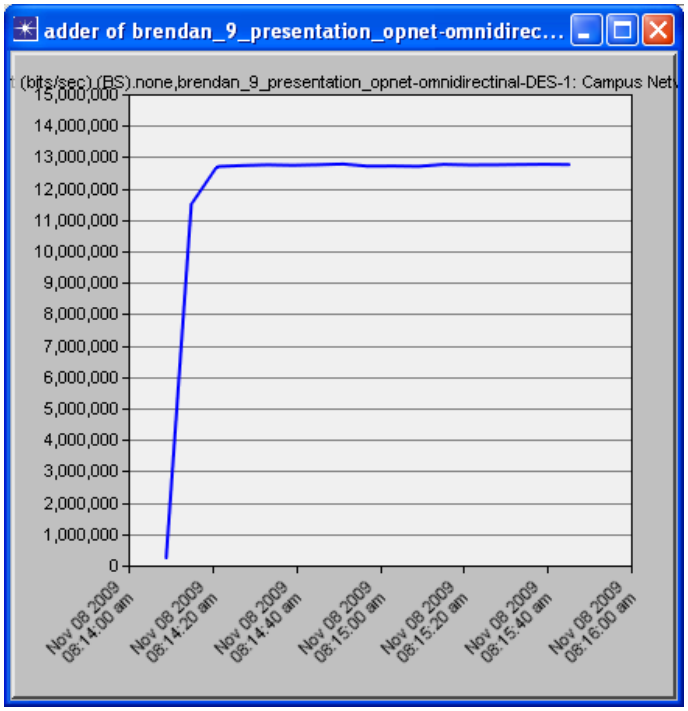

Figure 4.2 – sum of traffic sent by BS of relay station using omnidirectional antenna

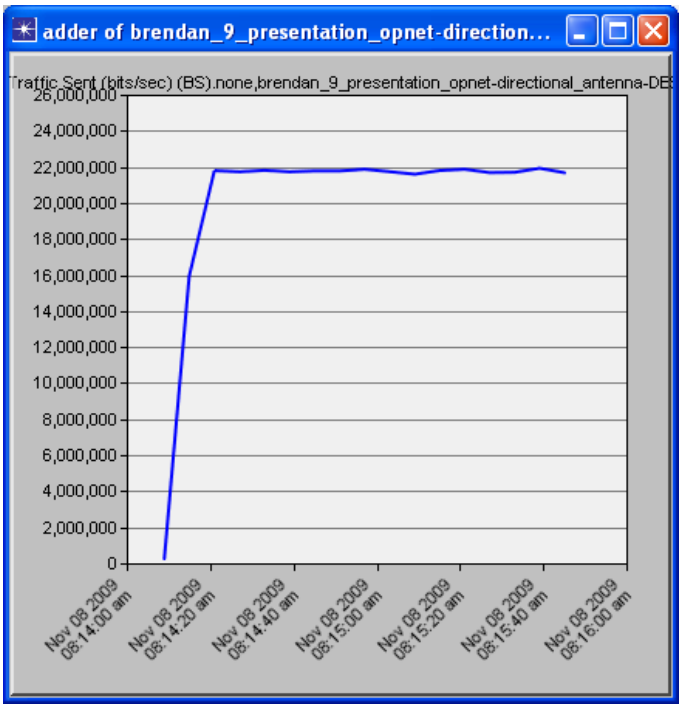

Figure 4.3 – sum of traffic sent by BS of relay station using directional antenna

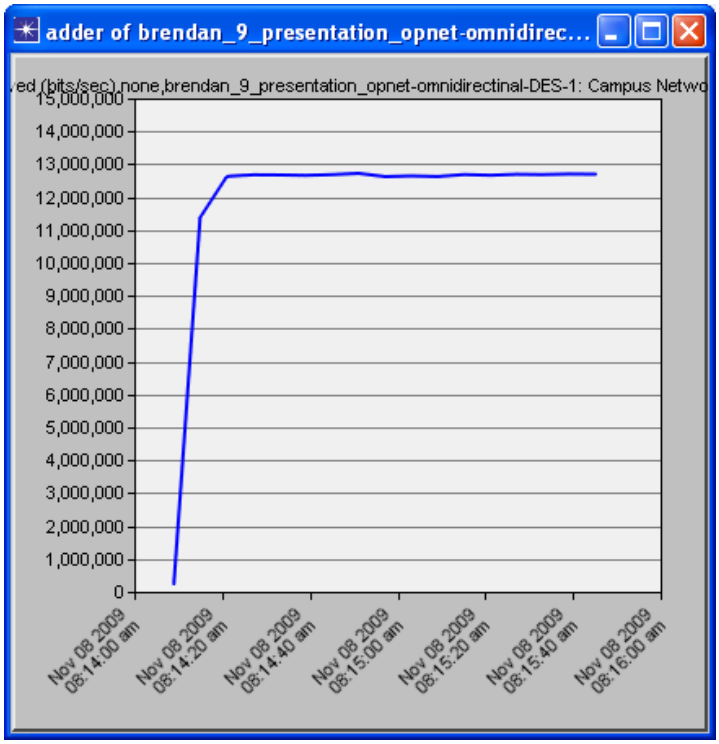

Figure 4.4 - sum of traffic sent by SS using omnidirectional antenna

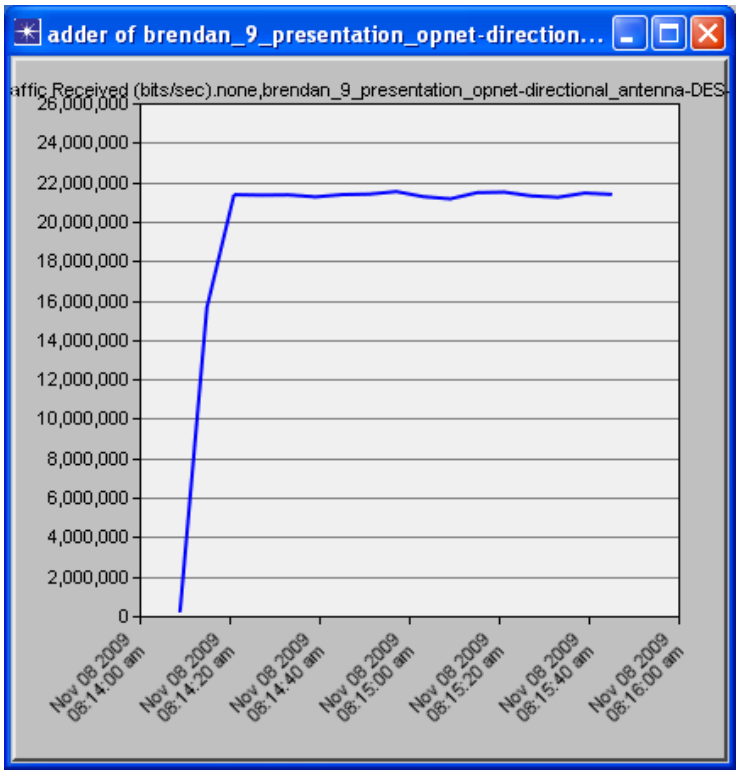

Figure 4.5 - sum of traffic sent by SS using directional antenna

Table 4.2 – Results comparing Simulation 1 and Simulation 2

Simulation 1	**Simulation 2**
Omni-directional antenna	directional antenna
Gain of BS and RS = 5dBi	Gain of BS and RS = 15dBi
Beam width = 360^{o}	Beam width = 40^{o}
Traffic sent by BS = 13 Mbps	Traffic sent by BS = 22.25 Mbps
Sum of Traffic sent by BS of relays = 12.8Mbps	Sum of Traffic sent by BS of relays = 22Mbps
Sum of Traffic sent by SS = 12.75 Mbps	Sum of Traffic sent by SS = 21.8 Mbps

From table 4.2, we can see that the throughput of the entire network is increased by 9 Mbps, when we use directional antennas as compared to omni-directional antennas.

Conclusion

We have automated the process of creating directional antennas in OPNET by using the EMA library. We have created a conical directional antenna, a cosine squared directional antenna and a pencil beam directional antenna. We have implemented tracking of a directional antenna which is extremely useful in mobile scenarios. When we combine the antenna creation and the tracking code, we can create and assign different antenna patterns to the base station as the subscriber stations are in motion. This is the

only way to use directional antenna in mobile cases. This algorithm can be used to create antenna patterns for any wireless technology in OPNET such as WiFi and WiMAX .Finally we created an ILP formulation for the RS-SS resource allocation problem which is proven to be NP-Complete. This is an optimization problem and the rounding Algorithm is a $\frac{1}{2\pi/\theta_{min}}$ – approximation algorithm in polynomial time. This resource allocation algorithm can be used in any technology that has a centralized resource allocation algorithm such as WiMAX, where the Base Station runs this algorithm. We have some simulation results showing 10Mbps improved throughput using directional antennas with the assignment given by our resource allocation algorithm.

REFERENCED CITED

REFERENCES

[1] Loutfi Nuaymi, “WiMAX Technology for Broadband Wireless Access”, *John Wiley and Sons, Inc., New York,* 2007.

[2] Constantine A. Balanis, “Antenna Theory Analysis and Design”, Third Edition, *John Wiley and Sons, Inc., New York,* 2005.

[3] Simon R. Saunders, “Antennas and Propagation For Wireless Communication Systems”, *Wiley and Sons, Inc., New York,* 2004.

[4] E Balagurusamy, “C Programming and Data Structures”, Tata McGraw-Hill Publishing Company Limited, *New Delhi* 2006.

[5] Michael Sullivan, “Algebra and Trignomentry”,2004.

[6] Mumey, Tang, Gurdasani, Wan, Wolff ,”Maximizing Throughput Relay Beam Steering Algorithm”, 2009,unpublished

APPENDICES

APPENDIX A

ANTENNA PATTERN CREATION AND TRACKING CODE IN OPNET

```
void
wimax_phy_mcarrier_pk_send (Packet* pkptr, WrlsT_Phy_Chnl_Info* phy_info_ptr, WrlsT_Phy_Mcarrier_Tx_Mgmt*
mcarrier_tx_ptr,
		WrlsT_Phy_Mcarrier_Burst_Info* busrt_alloc_info_ptr, Objid tx_module_objid)
		{
		int output_counduit_index;

		/*   ********************************************code added in summer fo 2009 by
neeraj**********************************/
		/*****************************defining variables in summer 2009**********************/

		double lsdf;
		double leh;
		double erg;
		double qwer;
		double dfg;
		double ert;
		double drg;



Objid tx_node_id;
Objid subnet_id;
Objid rx_node_id;
//Objid ant_node_id;
Objid ant_node_id1;
Objid ant_node_id2;
Objid rx_node_id11;
Objid rx_node_id12;
Objid rx_node_id13;
Objid rx_node_id14;
Objid rx_node_id15;
Objid rx_node_id16;
double x_pos;
double y_pos;
double z_pos;
double latitude;
double longitude;
double altitude;

double BS_lat;
double BS_log;
double BS_alt;
double BS1_lat;
double BS1_log;
double BS1_alt;
double BS2_lat;
double BS2_log;
double BS2_alt;
double BS3_lat;
double BS3_log;
double BS3_alt;
double m1;
double m2;
double theeta;
double theta;
double distance_a;
		double distance_b;
		double distance_c;
		double distance_a1;
		double distance_b1;
		double distance_c1;
		double distance_a2;
		double distance_b2;
```

```
	double distance_c2;
	double distance_a3;
	double distance_b3;
	double distance_c3;
	double distance_x;
	double distance_com;

	double distance_y;
	double distance_z1;
	double distance_z2;
	double distance_xalpha;
	double distance_yalpha;
	double radius;
	double theta_t1;
	double theta_t2;
	double theta_t3;
	double theta_phi;
	double theta1;
	double theta2;

	int
i,p_min_1,p_max_1,p_min_2,p_max_2,p_min_3,p_max_3,p_min_i,p_max_i,phi_min_1,phi_max_1,phi_min_2,phi_max_2,phi_min_
3,phi_max_3,phi_min_i,phi_max_i;
	int
theta_min_int,theta_max_int,phi_min_int_1,phi_max_int_1,phi_min_int_2,phi_max_int_2,phi_min_int_3,phi_max_int_3,phi_min_int
_i,phi_max_int_i,g,number_ss;
	int t=0;
	int aabb=0,phi_max[10],xy=0,phi_max__int[10],abcd=0;
	char c;
	FILE *f,*fp;
	FILE *f1;
	char s,filename[20];



	/** Inputs:                                                                    **/
	/**                     pkptr-> Packet to be send through the multicarrier PHY.  **/
	/**                     mcarrier_tx_ptr-> Multicarrier tx object.
	**/
	/**          phy_info_ptr-> Contains the configuration of the PHY layer.          **/
	/**                                                  (used by the wrls pipeline stages)
	**/
	/**                     Burst allocation
	                                **/
	/**                             subchnl_start-> First subchannel in the burt allocation    **/
	/**                             subchnl_count-> Total number of subchnls in the allocation **/
	/**                             end_of_txmission-> Time when the allocation ends           **/
	/**
	                                                **/
	/**                     Tx module objid
	                                        **/
	/**
	                                                        **/
	/** This function delivers a packet to the wrls pipeline stages (via   **/
	/** radio channel). It allows packets to be transmitted in parallel          **/
	/** as in a multicarrier system (e.g. OFDMA). "Conduits" are use to send**/
	/** the packets in "parallel" (e.g. because time overlap).                        **/
	/** Before a packet is sent a free conduit is located and reserved for       **/
	/** the entire transmission of a given burst.                                  **/
	/** The information about the burst allocation and the PHY layer             **/
	/** configuration are included in the packet in indexed fields that are      **/
	/** created "on the fly". Once the packet is sent, this framework            **/
	/** assumes that no more indexed fields will be added into this packet,      **/
	/** otherwise access to the burst information and PHY layer information      **/
	/** from the wrls pipeline stages will fail.                                   **/
	FIN (wimax_phy_mcarrier_pk_send (<args>));
```

```
		/* Set the burst allocation information in the packet.			*/
		op_pk_fd_set_ptr (pkptr, mac_access_burst_info_field_index(pkptr), busrt_alloc_info_ptr,
			0 /*field size zero*/, wrls_burst_info_copy_proc, op_prg_mem_free, sizeof
(WrlsT_Phy_Mcarrier_Burst_Info));

		/* Set the PHY channel information associated with this		*/
		/* transmission into the packet.                                                */
		op_pk_fd_set_ptr (pkptr, mac_access_phy_info_field_index(pkptr), phy_info_ptr,
			0 /*field size zero*/, wimax_phy_info_copy_proc, wimax_phy_info_destroy_proc, sizeof
(WrlsT_Phy_Chnl_Info));

		/* Obtain the next available conduit in the mcarrier tx.   */
		output_counduit_index = wimax_phy_mcarrier_tx_conduit_index_get (mcarrier_tx_ptr, op_sim_time () +
busrt_alloc_info_ptr->start_time,
			op_sim_time () + busrt_alloc_info_ptr->start_time + busrt_alloc_info_ptr->tx_delay);

		/* Deliver the packet to the radio tx module, use the next		*/
		/* available conduit for this transmission.                                     */

		/* ******************  summer of 2009- this is where the actual code is implemented**************************/

		//printf("plzzzz  %d fix \n",OPC_OBJTYPE_NODE_FIX);
		//printf("plzzzz  %d mob\n",OPC_OBJTYPE_NODE_MOB);

		tx_node_id = op_topo_parent (op_id_self ());
		subnet_id = op_topo_parent (tx_node_id);

		if(op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_FIX, "BS")==3)
		{
			//printf("ok on BS \n");
			tx_node_id = op_topo_parent (op_id_self ());
			subnet_id = op_topo_parent (tx_node_id);

		//ant_node_id1 = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_FIX, "BS");
		//ant_node_id2 = op_id_from_name (ant_node_id1, OPC_OBJTYPE_ANT, "wimax_ant_8_0");
		//rx_node_id = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_MOB, "SS");
		//op_ima_obj_pos_get (rx_node_id, &latitude, &longitude,&altitude, &x_pos, &y_pos, &z_pos);

		//************************************latitude
longitude*********************************************************************

		rx_node_id11 = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_FIX, "BS");
		op_ima_obj_pos_get (rx_node_id11, &latitude, &longitude,&altitude, &x_pos, &y_pos, &z_pos);
		BS_lat=latitude;
		BS_log=longitude;
		BS_alt=altitude;



		rx_node_id12 = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_MOB, "SS");
		op_ima_obj_pos_get (rx_node_id12, &latitude, &longitude,&altitude, &x_pos, &y_pos, &z_pos);
		BS1_lat=latitude;
		BS1_log=longitude;
		BS1_alt=altitude;

		//printf("%lf  %lf  %lf\n", BS1_log, BS1_lat,BS1_alt );


/*
		rx_node_id13 = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_MOB, "SS_2");
		op_ima_obj_pos_get (rx_node_id13, &latitude, &longitude,&altitude, &x_pos, &y_pos, &z_pos);
		BS2_lat=latitude;
		BS2_log=longitude;
		BS2_alt=altitude;
```

```
		rx_node_id14 = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_MOB, "SS_3");
		op_ima_obj_pos_get (rx_node_id14, &latitude, &longitude,&altitude, &x_pos, &y_pos, &z_pos);
		BS3_lat=latitude;
		BS3_log=longitude;
		BS3_alt=altitude;
		*/



		//rx_node_id15 = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_MOB, "SS4");
		//op_ima_obj_pos_get (rx_node_id15, &latitude, &longitude,&altitude, &x_pos, &y_pos, &z_pos);


		//rx_node_id16 = op_id_from_name (subnet_id, OPC_OBJTYPE_NODE_MOB, "SS5");
		//op_ima_obj_pos_get (rx_node_id16, &latitude, &longitude,&altitude, &x_pos, &y_pos, &z_pos);



		//*************************************harshas theeta calculations
start*************************************//

		/*
		distance_a = sqrt(((BS2_lat-BS_lat)*(BS2_lat-BS_lat)) + ((BS2_log-BS_log)*(BS2_log-BS_log)));
		distance_b = sqrt(((BS2_lat-BS1_lat)*(BS2_lat-BS1_lat)) + ((BS2_log-BS1_log)*(BS2_log-BS1_log)));
		distance_c = sqrt(((BS1_lat-BS_lat)*(BS1_lat-BS_lat)) + ((BS1_log-BS_log)*(BS1_log-BS_log)));

		theta = acos (((distance_c*distance_c) + (distance_a*distance_a) - (distance_b*distance_b))/(2*distance_a*distance_c));
		theta1=theta*180/3.14;


		distance_a = sqrt(((BS3_lat-BS_lat)*(BS3_lat-BS_lat)) + ((BS3_log-BS_log)*(BS3_log-BS_log)));
		distance_b = sqrt(((BS3_lat-BS1_lat)*(BS3_lat-BS1_lat)) + ((BS3_log-BS1_log)*(BS3_log-BS1_log)));
		distance_c = sqrt(((BS1_lat-BS_lat)*(BS1_lat-BS_lat)) + ((BS1_log-BS_log)*(BS1_log-BS_log)));

		theta = acos (((distance_c*distance_c) + (distance_a*distance_a) - (distance_b*distance_b))/(2*distance_a*distance_c));
		theta2=theta*180/3.14;



		just = 0;
		*/
		//just = 1;
		//just = 0;
		just = 1;


		//*************************************harshas theeta calculations end*************************************//
		/*************************************************************************************************************
**********************
		/*************************************************************************************************************
**********************
		/*************************************************************************************************************
**********************
		/*************************************************************************************************************
**********************/

		if(just==0)
		{
			printf("ohh no");
			just=just+1;


		f=fopen("abcdefg.em.c","w+");

		fputs("#include <opnet.h>",f);
```

```
		fputc('\n',f);
		fputs("#include <ema.h>",f);
		fputc('\n',f);
		fputs("#include <opnet_emadefs.h>",f);
		fputc('\n',f);
		fputs("#include <opnet_constants.h>",f);
		fputc('\n',f);
		fputc('\n',f);
		fputc('\n',f);



		//printf("enter the number of ss's\n");
//		scanf("%d",&number_ss);
		number_ss=3;

		for(aabb=0;aabb<number_ss-1;aabb=aabb+1)
		{
			//printf("enter the maximum value of phi angle of ss%d\n",aabb+2);
			//scanf("%d",&phi_max[xy]);
			phi_max[0]=(int)theta1;
			phi_max[1]=(int)theta2;
			phi_max__int[0]= (int) floor (phi_max[0]/5);
			phi_max__int[1]= (int) floor (phi_max[1]/5);
			xy=2;
		}



		/////printf("enter your gain \n");
		////scanf("%d",&g);
		g=22;
		//printf("enter the name of your antenna pattern (.pa file) without the .pa \n");
		//scanf("%s",filename);



//printf("ok1");
		for(i=0;i<36;i++)
		{


			if( (i==0)||(i==1)||(i==(phi_max__int[abcd]-1)) || (i==(phi_max__int[abcd])) || (i==(phi_max__int[abcd]+1)))
			{
				//printf("ok1");

			if(abcd<xy)
			{
				fputs("double",f);
		putc('\t',f);
		fputs("dvec_",f);
		fprintf (f, "%d",i);
		fputs(" [] =",f);
		putc('\n',f);
		putc('\t',f);
		putc('{',f);
		putc('\n',f);



		if((i==0)||(i==1))
		{
			for(t=0;t<72;t++)
			  {
				fprintf (f, "%d",g);
```

```
                        fprintf (f, "%s",", ");
                }
}
else
{
                for(t=0;t<72;t++)
                 {


                        if(t==70 || t==71)
                        {
                                fprintf (f, "%d",g);
                        }
                        else
                        {
                         fprintf (f, "%d",0);
                        }


                                                fprintf (f, "%s",", ");

                 }
}




                 putc('\n',f);
                 putc('\t',f);
                 putc('}',f);
                 putc(';',f);
                 putc('\n',f);
                 putc('\n',f);
                        if(i==(phi_max__int[abcd]+1))
                        abcd=abcd+1;
                }
                }



                /* interferance**********************
                else if(i>=phi_min_int_i && i<=phi_max_int_i)
                {
                        fun3(i,phi_min_int_i,phi_max_int_i,g,f);
                }
                *****************************************/
                else
                {
                        //printf("ok2");
                        //fun2(i,theta_min_int,theta_max_int,g,f);


                                                f1=fopen("abcd.txt","r");
//printf("%d",a);
                                                fputc('\n',f);
                                                fputs("double",f);
                                                putc('\t',f);
                                                fputs("dvec_",f);
                                                fprintf (f, "%d",i);
                                                fputs(" [] =",f);

                                        while( ( (c=getc(f1)) != EOF)   )
                                                {
                                                putc( c,f);
```

```
                                                        }
                                                        putc('\n',f);
                                                        putc('\n',f);
                                                        fclose(f1);
                        }


                }
//printf("ok2");
                fp=fopen("end.txt","r+");
//printf("ok3");



while( ( (s=getc(fp)) != EOF) )
{

                        putc(s,f);
                }
//printf("ok4");

                fclose(f);
                fclose(fp);
//printf("ok5");


                //opnen f3, rewind,put headders

                //(void) system("c:\>op_mkema -m abcdefg");
                // system("C:/Documents and Settings/neeraj.gurdasani/My Documents/op_models/abcdefg.dev32.i0.em.x");
                //(void) system ("c:/Documents and Settings/neeraj.gurdasani/My Documents/op_models/abcdefg.dev32.i0.em");
                //fp=fopen("abcdefg.dev32.i0.em.x","r+");
                //fclose(fp);
                //spawnl(P_WAIT  , "abcdefg.dev32.i0.em.x" , NULL);
                //Documents and Settings\neeraj.gurdasani\My Documents\op_models
                system("op_mkema -m abcdefg");
                system("abcdefg.dev32.i0.em.x");
                //system("c:","
                //system("cd Documents and Settings\neeraj.gurdasani\My Documents\op_models");
                //system(" op_mkema -m abcdefg");
                //ShellExecute(GetDesktopWindow(), "open", "c:\Documents and Settings\neeraj.gurdasani\My
Documents\op_models\op_mkema -m abcdefg", NULL, NULL, SW_SHOWNORMAL);
                //ShellExecute(NULL, NULL, "C:/Documents and Settings/neeraj.gurdasani/My
Documents/op_models/abcdefg.dev32.i0.em.x", NULL, NULL, SW_SHOWNORMAL);
//op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "pattern", "mont");
                //op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "target latitude", BS1_lat);
                //op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "target longitude", BS1_log);
                //op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "target altitude", BS1_alt);



}









                /*****************************************************************************************************
**********************
```

```
	****************************************************************************************************
********************

	****************************************************************************************************
********************

	****************************************************************************************************
********************/
//printf("ok implement");
	//op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "pattern", "cone_5");
	op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "target latitude", BS1_lat);
	op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "target longitude", BS1_log);
	op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "target altitude", BS1_alt);

	op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_MOB, "SS"), OPC_OBJTYPE_ANT, "wimax_ant_0_0"), "pattern", "cone_5");
	op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_MOB, "SS"), OPC_OBJTYPE_ANT, "wimax_ant_0_0"), "target latitude", BS_lat);
	op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_MOB, "SS"), OPC_OBJTYPE_ANT, "wimax_ant_0_0"), "target longitude", BS_log);
	op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_MOB, "SS"), OPC_OBJTYPE_ANT, "wimax_ant_0_0"), "target altitude", BS_alt);
//op_ima_obj_attr_set (op_id_from_name (op_id_from_name (op_topo_parent (op_topo_parent (op_id_self ())),
OPC_OBJTYPE_NODE_FIX, "BS_aac"), OPC_OBJTYPE_ANT, "wimax_ant_8_0"), "target latitude", latitude);
}
	op_pk_deliver_delayed (pkptr, tx_module_objid, output_counduit_index, busrt_alloc_info_ptr->start_time);

	FOUT
	}
```

Abcd.txt

```
	{
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000, -20.000000000000,
	-20.000000000000, -20.000000000000,
	};
```

End.txt

```
/* array for all objects in model */
EmaT_Object_Id			obj [36];



int
main (int argc, char* argv [])
	{
	EmaT_Model_Id				model_id;
	int						i;
```

```
/* initialize EMA package */
Ema_Init (EMAC_MODE_ERR_PRINT | EMAC_MODE_REL_60, argc, argv);

/* create an empty model */
model_id = Ema_Model_Create (MOD_ANT_PATTERN);



/* create all objects */
obj [0] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [1] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [2] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [3] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [4] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [5] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [6] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [7] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [8] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [9] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [10] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [11] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [12] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [13] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [14] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [15] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [16] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [17] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [18] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [19] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [20] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [21] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [22] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [23] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [24] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [25] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [26] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [27] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [28] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [29] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [30] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [31] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [32] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [33] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [34] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);
obj [35] = Ema_Object_Create (model_id, OBJ_PAT_PLANE);



/* set the model level attributes */
Ema_Model_Attr_Set (model_id,
                "num planes",          COMP_CONTENTS, 36,
                "plane array",         COMP_ARRAY_CONTENTS (0), obj [0],
                "plane array",         COMP_ARRAY_CONTENTS (1), obj [1],
                "plane array",         COMP_ARRAY_CONTENTS (2), obj [2],
                "plane array",         COMP_ARRAY_CONTENTS (3), obj [3],
                "plane array",         COMP_ARRAY_CONTENTS (4), obj [4],
                "plane array",         COMP_ARRAY_CONTENTS (5), obj [5],
                "plane array",         COMP_ARRAY_CONTENTS (6), obj [6],
                "plane array",         COMP_ARRAY_CONTENTS (7), obj [7],
                "plane array",         COMP_ARRAY_CONTENTS (8), obj [8],
                "plane array",         COMP_ARRAY_CONTENTS (9), obj [9],
                "plane array",         COMP_ARRAY_CONTENTS (10), obj [10],
                "plane array",         COMP_ARRAY_CONTENTS (11), obj [11],
                "plane array",         COMP_ARRAY_CONTENTS (12), obj [12],
                "plane array",         COMP_ARRAY_CONTENTS (13), obj [13],
```

```
                "plane array",      COMP_ARRAY_CONTENTS (14), obj [14],
                EMAC_EOL);

        Ema_Model_Attr_Set (model_id,
                "plane array",      COMP_ARRAY_CONTENTS (15), obj [15],
                "plane array",      COMP_ARRAY_CONTENTS (16), obj [16],
                "plane array",      COMP_ARRAY_CONTENTS (17), obj [17],
                "plane array",      COMP_ARRAY_CONTENTS (18), obj [18],
                "plane array",      COMP_ARRAY_CONTENTS (19), obj [19],
                "plane array",      COMP_ARRAY_CONTENTS (20), obj [20],
                "plane array",      COMP_ARRAY_CONTENTS (21), obj [21],
                "plane array",      COMP_ARRAY_CONTENTS (22), obj [22],
                "plane array",      COMP_ARRAY_CONTENTS (23), obj [23],
                "plane array",      COMP_ARRAY_CONTENTS (24), obj [24],
                "plane array",      COMP_ARRAY_CONTENTS (25), obj [25],
                "plane array",      COMP_ARRAY_CONTENTS (26), obj [26],
                "plane array",      COMP_ARRAY_CONTENTS (27), obj [27],
                "plane array",      COMP_ARRAY_CONTENTS (28), obj [28],
                "plane array",      COMP_ARRAY_CONTENTS (29), obj [29],
                "plane array",      COMP_ARRAY_CONTENTS (30), obj [30],
                EMAC_EOL);

        Ema_Model_Attr_Set (model_id,
                "plane array",      COMP_ARRAY_CONTENTS (31), obj [31],
                "plane array",      COMP_ARRAY_CONTENTS (32), obj [32],
                "plane array",      COMP_ARRAY_CONTENTS (33), obj [33],
                "plane array",      COMP_ARRAY_CONTENTS (34), obj [34],
                "plane array",      COMP_ARRAY_CONTENTS (35), obj [35],
                EMAC_EOL);



        /* assign attrs for object 'obj [0]' */
        Ema_Object_Attr_Set (model_id, obj [0],
                "view low gain",    COMP_CONTENTS, (double) -21,
                "view high gain",   COMP_CONTENTS, (double) -19,
                "gain vec",         COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [0], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [0], "gain vec", COMP_DVEC_CONTENTS(i), dvec_0 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [1]' */
        Ema_Object_Attr_Set (model_id, obj [1],
                "view low gain",    COMP_CONTENTS, (double) -21,
                "view high gain",   COMP_CONTENTS, (double) -19,
                "gain vec",         COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [1], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [1], "gain vec", COMP_DVEC_CONTENTS(i), dvec_1 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [2]' */
        Ema_Object_Attr_Set (model_id, obj [2],
                "view low gain",    COMP_CONTENTS, (double) -21,
```

```
                "view high gain",       COMP_CONTENTS, (double) -19,
                "gain vec",             COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [2], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [2], "gain vec", COMP_DVEC_CONTENTS(i), dvec_2 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [3]' */
        Ema_Object_Attr_Set (model_id, obj [3],
                "view low gain",        COMP_CONTENTS, (double) -21,
                "view high gain",       COMP_CONTENTS, (double) -19,
                "gain vec",             COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [3], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [3], "gain vec", COMP_DVEC_CONTENTS(i), dvec_3 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [4]' */
        Ema_Object_Attr_Set (model_id, obj [4],
                "view low gain",        COMP_CONTENTS, (double) -21,
                "view high gain",       COMP_CONTENTS, (double) -19,
                "gain vec",             COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [4], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [4], "gain vec", COMP_DVEC_CONTENTS(i), dvec_4 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [5]' */
        Ema_Object_Attr_Set (model_id, obj [5],
                "view low gain",        COMP_CONTENTS, (double) -21,
                "view high gain",       COMP_CONTENTS, (double) -19,
                "gain vec",             COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [5], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [5], "gain vec", COMP_DVEC_CONTENTS(i), dvec_5 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [6]' */
        Ema_Object_Attr_Set (model_id, obj [6],
                "view low gain",        COMP_CONTENTS, (double) -21,
                "view high gain",       COMP_CONTENTS, (double) -19,
                "gain vec",             COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);
```

```
	Ema_Object_Attr_Set (model_id, obj [6], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [6], "gain vec", COMP_DVEC_CONTENTS(i), dvec_6 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [7]' */
	Ema_Object_Attr_Set (model_id, obj [7],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [7], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [7], "gain vec", COMP_DVEC_CONTENTS(i), dvec_7 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [8]' */
	Ema_Object_Attr_Set (model_id, obj [8],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [8], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [8], "gain vec", COMP_DVEC_CONTENTS(i), dvec_8 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [9]' */
	Ema_Object_Attr_Set (model_id, obj [9],
		"view low gain",	COMP_CONTENTS, (double) 28.8124999999999,
		"view high gain",	COMP_CONTENTS, (double) 31.845394736842,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [9], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [9], "gain vec", COMP_DVEC_CONTENTS(i), dvec_9 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [10]' */
	Ema_Object_Attr_Set (model_id, obj [10],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [10], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);
```

```
	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [10], "gain vec", COMP_DVEC_CONTENTS(i), dvec_10 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [11]' */
	Ema_Object_Attr_Set (model_id, obj [11],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [11], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [11], "gain vec", COMP_DVEC_CONTENTS(i), dvec_11 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [12]' */
	Ema_Object_Attr_Set (model_id, obj [12],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [12], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [12], "gain vec", COMP_DVEC_CONTENTS(i), dvec_12 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [13]' */
	Ema_Object_Attr_Set (model_id, obj [13],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [13], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [13], "gain vec", COMP_DVEC_CONTENTS(i), dvec_13 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [14]' */
	Ema_Object_Attr_Set (model_id, obj [14],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [14], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
```

```
		Ema_Object_Attr_Set (model_id, obj [14], "gain vec", COMP_DVEC_CONTENTS(i), dvec_14 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [15]' */
	Ema_Object_Attr_Set (model_id, obj [15],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [15], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [15], "gain vec", COMP_DVEC_CONTENTS(i), dvec_15 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [16]' */
	Ema_Object_Attr_Set (model_id, obj [16],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [16], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [16], "gain vec", COMP_DVEC_CONTENTS(i), dvec_16 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [17]' */
	Ema_Object_Attr_Set (model_id, obj [17],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [17], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [17], "gain vec", COMP_DVEC_CONTENTS(i), dvec_17 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [18]' */
	Ema_Object_Attr_Set (model_id, obj [18],
		"view low gain",	COMP_CONTENTS, (double) 19.3333333333329,
		"view high gain",	COMP_CONTENTS, (double) 21.3684210526311,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [18], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [18], "gain vec", COMP_DVEC_CONTENTS(i), dvec_18 [i],
EMAC_EOL);
		}
```

```
	/* assign attrs for object 'obj [19]' */
	Ema_Object_Attr_Set (model_id, obj [19],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [19], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [19], "gain vec", COMP_DVEC_CONTENTS(i), dvec_19 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [20]' */
	Ema_Object_Attr_Set (model_id, obj [20],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [20], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [20], "gain vec", COMP_DVEC_CONTENTS(i), dvec_20 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [21]' */
	Ema_Object_Attr_Set (model_id, obj [21],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [21], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [21], "gain vec", COMP_DVEC_CONTENTS(i), dvec_21 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [22]' */
	Ema_Object_Attr_Set (model_id, obj [22],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [22], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [22], "gain vec", COMP_DVEC_CONTENTS(i), dvec_22 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [23]' */
```

```
	Ema_Object_Attr_Set (model_id, obj [23],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [23], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [23], "gain vec", COMP_DVEC_CONTENTS(i), dvec_23 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [24]' */
	Ema_Object_Attr_Set (model_id, obj [24],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [24], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [24], "gain vec", COMP_DVEC_CONTENTS(i), dvec_24 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [25]' */
	Ema_Object_Attr_Set (model_id, obj [25],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [25], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [25], "gain vec", COMP_DVEC_CONTENTS(i), dvec_25 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [26]' */
	Ema_Object_Attr_Set (model_id, obj [26],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [26], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [26], "gain vec", COMP_DVEC_CONTENTS(i), dvec_26 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [27]' */
	Ema_Object_Attr_Set (model_id, obj [27],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
```

```
                "gain vec",         COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [27], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [27], "gain vec", COMP_DVEC_CONTENTS(i), dvec_27 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [28]' */
        Ema_Object_Attr_Set (model_id, obj [28],
                "view low gain",    COMP_CONTENTS, (double) -21,
                "view high gain",   COMP_CONTENTS, (double) -19,
                "gain vec",         COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [28], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [28], "gain vec", COMP_DVEC_CONTENTS(i), dvec_28 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [29]' */
        Ema_Object_Attr_Set (model_id, obj [29],
                "view low gain",    COMP_CONTENTS, (double) -21,
                "view high gain",   COMP_CONTENTS, (double) -19,
                "gain vec",         COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [29], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [29], "gain vec", COMP_DVEC_CONTENTS(i), dvec_29 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [30]' */
        Ema_Object_Attr_Set (model_id, obj [30],
                "view low gain",    COMP_CONTENTS, (double) -21,
                "view high gain",   COMP_CONTENTS, (double) -19,
                "gain vec",         COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);

        Ema_Object_Attr_Set (model_id, obj [30], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

        for (i = 0; i < 72; i++)
                {
                Ema_Object_Attr_Set (model_id, obj [30], "gain vec", COMP_DVEC_CONTENTS(i), dvec_30 [i],
EMAC_EOL);
                }


        /* assign attrs for object 'obj [31]' */
        Ema_Object_Attr_Set (model_id, obj [31],
                "view low gain",    COMP_CONTENTS, (double) -21,
                "view high gain",   COMP_CONTENTS, (double) -19,
                "gain vec",         COMP_INTENDED, EMAC_DISABLED,
                EMAC_EOL);
```

```
	Ema_Object_Attr_Set (model_id, obj [31], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [31], "gain vec", COMP_DVEC_CONTENTS(i), dvec_31 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [32]' */
	Ema_Object_Attr_Set (model_id, obj [32],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [32], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [32], "gain vec", COMP_DVEC_CONTENTS(i), dvec_32 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [33]' */
	Ema_Object_Attr_Set (model_id, obj [33],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [33], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [33], "gain vec", COMP_DVEC_CONTENTS(i), dvec_33 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [34]' */
	Ema_Object_Attr_Set (model_id, obj [34],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [34], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
		{
		Ema_Object_Attr_Set (model_id, obj [34], "gain vec", COMP_DVEC_CONTENTS(i), dvec_34 [i],
EMAC_EOL);
		}


	/* assign attrs for object 'obj [35]' */
	Ema_Object_Attr_Set (model_id, obj [35],
		"view low gain",	COMP_CONTENTS, (double) -21,
		"view high gain",	COMP_CONTENTS, (double) -19,
		"gain vec",		COMP_INTENDED, EMAC_DISABLED,
		EMAC_EOL);

	Ema_Object_Attr_Set (model_id, obj [35], "gain vec", COMP_DVEC_SIZE, 72, EMAC_EOL);

	for (i = 0; i < 72; i++)
```

```
                {
                Ema_Object_Attr_Set (model_id, obj [35], "gain vec", COMP_DVEC_CONTENTS(i), dvec_35 [i],
EMAC_EOL);
                }
```

```
/* write the model to application-readable form */
Ema_Model_Write (model_id, "mont");

return 0;
}
```

APPENDIX B

COSINE SQUARED GAIN ANTENNA PATTERN CREATION IN OPNET

```
// antenna_hard.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>
#include <math.h>
#include <string.h>
#define MAX_LEN_SINGLE_LINE 120
#define PI 3.14159265


int main()
{
	int i,theta_min,theta_max,phi_min,phi_max;
	int theta_min_int,theta_max_int,phi_min_int,phi_max_int,g;

	FILE *f,*fp;
	char s,filename[20];
	f=fopen("abcde.em.c","w+");

	fputs("#include <opnet.h>",f);
	fputc('\n',f);
	fputs("#include <ema.h>",f);
	fputc('\n',f);
	fputs("#include <opnet_emadefs.h>",f);
	fputc('\n',f);
	fputs("#include <opnet_constants.h>",f);
	fputc('\n',f);
	fputc('\n',f);
	fputc('\n',f);



	printf("enter the minimum value of theta angle\n");
	scanf("%d",&theta_min);
	printf("enter the maximum value of theta angle\n");
	scanf("%d",&theta_max);
	printf("enter the minimum value of phi angle\n");
	scanf("%d",&phi_min);
	printf("enter the maximum value of phi angle\n");
	scanf("%d",&phi_max);
	printf("enter your gain \n");
	scanf("%d",&g);
	printf("enter the name of your antenna pattern (.pa file) without the .pa \n");
	scanf("%s",filename);

	theta_min_int=(theta_min/5);
	theta_max_int=(theta_max/5);
	phi_min_int=(phi_min/5);
	phi_max_int=(phi_max/5);

printf("ok1");
	for(i=0;i<36;i++)
	{
		if(i<phi_min_int || i>=phi_max_int )
		{
			fun1(i,f);
		}
		else
		{
			fun2(i,theta_min_int,theta_max_int,g,f);
		}
```

```
		}
printf("ok2");
		fp=fopen("end.txt","r");
printf("ok3");



while( ( (s=getc(fp)) != EOF) )
{

			putc(s,f);
		}
printf("ok4");

		fclose(f);
		fclose(fp);
printf("ok5");


		//opnen f3, rewind,put headders

		return 0;
}


fun1(a,f2)
{

		FILE *f1;
		char c;
		f1=fopen("abcd.txt","r");
		//printf("%d",a);
		fputc('\n',f1);
		fputs("double",f2);
		putc('\t',f2);
		fputs("dvec_",f2);
		fprintf (f2, "%d",a);
		fputs(" [] =",f2);

		while( ( (c=getc(f1)) != EOF)   )
		{
			putc( c,f2);
		}
		putc('\n',f2);
		putc('\n',f2);
		fclose(f1);

}

fun2(b,tmin,tmax,gain,f4)
{
		FILE *f3;
		int t,q=-1;
		int d,x7;
		f3=fopen("abcde.txt","r+");

		fputs("double",f4);
		putc('\t',f4);
		fputs("dvec_",f4);
		fprintf (f4, "%d",b);
		fputs(" [] =",f4);
		putc('\n',f4);
		putc('\t',f4);
		putc('{',f4);
		putc('\n',f4);

		d= floor((tmax-tmin)/2);
```

```
		//printf("%d\n",d);
		 x7=-d;

		for(t=0;t<72;t++)
				{

					q=q+1;
					if(q<tmin || q>=tmax)
					{
						fputs("-20.000000000000, ",f4);
					}
					else
					{
					// printf("\n");
					 //printf("%d is x7\n",x7);
					 //printf("%d is d\n",d);
					 //printf("%f\n", cos (((float)x7/d)*(90*PI/180)) );
					 //printf("\n");
					 x7++;


					 //putw(  cos (((float)x7/d)*(90*PI/180)) * gain , f4) ;
						fprintf (f4, "%f",((cos (  ((float)x7/d *(90*PI/180)) * ((float)x7/d
*(90*PI/180)) ) ) ) * gain);

						fprintf (f4, "%s",", ");

/*
					 fputs("4.000000000000, ",f4);
*/
					}


			}

			putc('\n',f4);
			putc('\t',f4);
			putc('}',f4);
			putc(';',f4);
			putc('\n',f4);
			putc('\n',f4);

			fclose(f3);
}
```

APPENDIX C

PENCIL BEAM ANTENNA PATTERN CTEATION CODE IN OPNET

```
// antenna_hard_2010.cpp : Defines the entry point for the console application.
//

// antenna_hard.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>
#include <math.h>
#include <string.h>
#define MAX_LEN_SINGLE_LINE 120
#define PI 3.14159265


int main()
{
	int
i,p_min_1,p_max_1,p_min_2,p_max_2,p_min_3,p_max_3,p_min_i,p_max_i,phi_min_1,phi_max_1,phi_min_2,phi_max_2,phi_min_
3,phi_max_3,phi_min_i,phi_max_i;
	int
theta_min_int,theta_max_int,phi_min_int_1,phi_max_int_1,phi_min_int_2,phi_max_int_2,phi_min_int_3,phi_max_int_3,phi_min_int
_i,phi_max_int_i,g,number_ss;

	int aabb=0,phi_max[10],xy=0,phi_max__int[10],abcd=0;

	FILE *f,*fp;
	char s,filename[20];
	f=fopen("abcdefg.em.c","w+");

	fputs("#include <opnet.h>",f);
	fputc('\n',f);
	fputs("#include <ema.h>",f);
	fputc('\n',f);
	fputs("#include <opnet_emadefs.h>",f);
	fputc('\n',f);
	fputs("#include <opnet_constants.h>",f);
	fputc('\n',f);
	fputc('\n',f);
	fputc('\n',f);



	//printf("enter the number of ss's\n");
//	scanf("%d",&number_ss);
	number_ss=3;

	for(aabb=0;aabb<number_ss-1;aabb=aabb+1)
	{
		printf("enter the maximum value of phi angle of ss%d\n",aabb+2);
		scanf("%d",&phi_max[xy]);
		phi_max__int[xy]= (int) floor (phi_max[xy]/5);
		xy=xy+1;
	}



	printf("enter your gain \n");
	scanf("%d",&g);
	//printf("enter the name of your antenna pattern (.pa file) without the .pa \n");
	//scanf("%s",filename);
```

```
//printf("ok1");
		for(i=0;i<36;i++)
		{




			if( (i==0)||(i==1)||(i==(phi_max__int[abcd]-1)) || (i==(phi_max__int[abcd])) || (i==(phi_max__int[abcd]+1)))
			{
				//printf("ok1");

if(abcd<xy)
			{
				fun2(i,g,f);
				if(i==(phi_max__int[abcd]+1))
				abcd=abcd+1;
			}
			}



			/* interferance**********************
			else if(i>=phi_min_int_i && i<=phi_max_int_i)
			{
				fun3(i,phi_min_int_i,phi_max_int_i,g,f);
			}
			*********************************************/
			else
			{
				//printf("ok2");
				//fun2(i,theta_min_int,theta_max_int,g,f);
				fun1(i,f);
			}


		}
//printf("ok2");
		fp=fopen("end.txt","r");
//printf("ok3");



while( ( (s=getc(fp)) != EOF) )
{
			putc(s,f);
		}
//printf("ok4");

		fclose(f);
		fclose(fp);
//printf("ok5");


		//opnen f3, rewind,put headders

		return 0;
}
```

```
fun1(a,f2)
{

	FILE *f1;
	char c;
	f1=fopen("abcd.txt","r");
	//printf("%d",a);
	fputc('\n',f2);
	fputs("double",f2);
	putc('\t',f2);
	fputs("dvec_",f2);
	fprintf (f2, "%d",a);
	fputs(" [] =",f2);

	while( ( (c=getc(f1)) != EOF)  )
	{
		putc( c,f2);
	}
	putc('\n',f2);
	putc('\n',f2);
	fclose(f1);

}

fun2(b,gain,f4)
{
	//FILE *f3;
	int t,q=-1;
	int d,x7;
//	f3=fopen("abcde.txt","r+");

	fputs("double",f4);
	putc('\t',f4);
	fputs("dvec_",f4);
	fprintf (f4, "%d",b);
	fputs(" [] =",f4);
	putc('\n',f4);
	putc('\t',f4);
	putc('{',f4);
	putc('\n',f4);

	if(  (b==0) || (b==1) )
	{
		for(t=0;t<72;t++)
		  {
			fprintf (f4, "%d",gain);
			fprintf (f4, "%s",", ");
		}
	}
	else
	{

		for(t=0;t<72;t++)
		  {


			if(t==70 || t==71)
			{
				fprintf (f4, "%d",gain);
			}
			else
			{
			  fprintf (f4, "%d",0);
			}
```

```
                                fprintf (f4, "%s",", ");


                        }
                }
        }


                        putc('\n',f4);
                        putc('\t',f4);
                        putc('}',f4);
                        putc(';',f4);
                        putc('\n',f4);
                        putc('\n',f4);

                       // fclose(f3);
}

fun3(b,p_min_i,p_max_i,gain,f4)
{
        //FILE *f3;
        int t,q=-1;
        int d,x7;
//      f3=fopen("abcde.txt","r+");

        fputs("double",f4);
        putc('\t',f4);
        fputs("dvec_",f4);
        fprintf (f4, "%d",b);
        fputs(" [] =",f4);
        putc('\n',f4);
        putc('\t',f4);
        putc('{',f4);
        putc('\n',f4);

        ////d= floor((tmax-tmin)/2);
        //printf("%d\n",d);
         ////x7=-d;
        //printf("min and%d\n",p_min_i);
        //printf("min and%d\n",p_max_i);

        //q++;


                for(t=0;t<72;t++)
                 {

                        ////  q=q+1;
                         ////if(q>=p_min_i && q<=p_max_i)
                        //// {
                                // printf("ok88\n");
                        ////      fputs("-20.000000000000, ",f4);

                         //// }

                         ////else if((q>=p_min_1 && q<=p_max_1) || (q>=p_min_2 && q<=p_max_2) ||(q>=p_min_3 &&
q<=p_max_3))
                          ////{
                                //  printf("%d\n",gain);
                                 // printf("\n");
                                  //printf("%d is x7\n",x7);
```

```
                                        //printf("%d is d\n",d);
                                        //printf("%f\n", cos (((float)x7/d)*(90*PI/180)) );
                                        //printf("\n");
                                       //  x7++;


                                        //putw(  cos (((float)x7/d)*(90*PI/180)) * gain , f4) ;
                                       //              fprintf (f4, "%f",((cos (  ((float)x7/d *(90*PI/180)) * ((float)x7/d
*(90*PI/180)) ) ) ) * gain);
                                       //  fprintf (f4, "%d",gain);
                                       //              fprintf (f4, "%s",", ");


                                        fputs("-20.000000000000, ",f4);



                                   //// }
                                    ////else
                                    ////{
                                                 ////   fputs("0.000000000000, ",f4);
                                    ////}
                                   //  q++;


                          }



                          putc('\n',f4);
                          putc('\t',f4);
                          putc('}',f4);
                          putc(';',f4);
                          putc('\n',f4);
                          putc('\n',f4);

                         // fclose(f3);
}
```

APPENDIX D

ILP FORMULATION FOR OPNET COORDINATES

```
// brendan_alg_5.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include<math.h>
#include <stdio.h>
#include <stdlib.h>

#define no_of_RS 7
#define no_of_SS 30

int main()
{

		//int RS_num = no_of_RS-1;
		int l=0;
		int k=0;
		int la=0;
		int ne = 0;
		int br=0;
		int rw = 0;
		int nt=0;
		int i,j=0;
		int b;
		int nu=0;
		 int a=0;
		 int z=0,y=0,x;
		 int rr=0;
		 int aa=0,bb=0;
		 int cc=0;
		 int dd=0;
		 int alp=0;
		 int bet=0;
		 float dis_x=0;
		 float dis_y=0;
		 int gm=0;
		 int mac_cc=0;
		 int S[no_of_SS][no_of_RS];
		 int temp = 0;
		 int flag1[no_of_SS];

		 FILE *f1;


		float RS_x[no_of_RS];
		float RS_y[no_of_RS];
		float angle_frm_fun[no_of_SS];

		int Beam_sets[no_of_RS][100][100];
		int beamset_temp[no_of_RS][100][100];
		int max_third[100][100];

		float s_sets[no_of_SS];
		float BS[no_of_RS][no_of_SS];

		//int SS_num = no_of_SS-1;
		float SS_x[no_of_SS];
```

```
float SS_y[no_of_SS];

float fun_angle();
//assignment of RS

f1=fopen("var.txt","w");

 RS_x[0]=2.056*200;
RS_y[0]=(5-2.88)*200;

RS_x[1]=1.92*200;
RS_y[1]=(5-2.41)*200;

RS_x[2]=2.096*200;
RS_y[2]=(5-1.986)*200;

RS_x[3]=2.526*200;
RS_y[3]=(5-1.882)*200;

RS_x[4]=2.445*200;
RS_y[4]=(5-2.973)*200;

RS_x[5]=2.875*200;
RS_y[5]=(5-2.95)*200;

RS_x[6]=3.24*200;
RS_y[6]=(5-2.84)*200;

// RS_x[1]=500;
// RS_y[1]=500;

//  RS_x[1]=100;
// RS_y[1]=120;

 //assifnment of ss
/*
 SS_x[0]=250;
 SS_y[0]=250;

 SS_x[1]=250;
 SS_y[1]=250;

 SS_x[2]=250;
 SS_y[2]=750;

 SS_x[3]=251;
 SS_y[3]=751;

 SS_x[4]=750;
 SS_y[4]=750;

 SS_x[5]=750;
 SS_y[5]=751;

 SS_x[6]=750;
 SS_y[6]=250;

 SS_x[7]=751;
 SS_y[7]=251;
 */

SS_x[0]=0.604*200;
SS_y[0]=(5-3.304)*200;

SS_x[1]=0.563*200;
SS_y[1]=(5-2.96)*200;
```

```
SS_x[2]=0.581*200;
SS_y[2]=(5-2.62)*200;

SS_x[3]=0.546*200;
SS_y[3]=(5-2.27)*200;

SS_x[4]=0.639*200;
SS_y[4]=(5-2.09)*200;

SS_x[5]=0.6911*200;
SS_y[5]=(5-1.835)*200;

SS_x[6]=0.685*200;
SS_y[6]=(5-1.5)*200;

SS_x[7]=0.883*200;
SS_y[7]=(5-1.35)*200;

SS_x[8]=1.07*200;
SS_y[8]=(5-1.18)*200;

SS_x[9]=1.3*200;
SS_y[9]=(5-1.1)*200;

SS_x[10]=1.55*200;
SS_y[10]=(5-0.952)*200;

SS_x[11]=1.81*200;
SS_y[11]=(5-0.877)*200;

SS_x[12]=2.12*200;
SS_y[12]=(5-0.842)*200;

SS_x[13]=2.387*200;
SS_y[13]=(5-0.7724)*200;

SS_x[14]=4.129*200;
SS_y[14]=(5-3.345)*200;

SS_x[15]=4.18*200;
SS_y[15]=(5-3.69)*200;

SS_x[16]=3.972*200;
SS_y[16]=(5-3.955)*200;

SS_x[17]=3.728*200;
SS_y[17]=(5-4.135)*200;

SS_x[18]=3.51*200;
SS_y[18]=(5-4.26)*200;

SS_x[19]=3.299*200;
SS_y[19]=(5-4.367)*200;

SS_x[20]=3.04*200;
SS_y[20]=(5-4.28)*200;

SS_x[21]=2.776*200;
SS_y[21]=(5-4.315)*200;

SS_x[22]=2.364*200;
SS_y[22]=(5-4.547)*200;

SS_x[23]=1.986*200;
SS_y[23]=(5-4.53)*200;

SS_x[24]=1.742*200;
```

```
SS_y[24]=(5-4.437)*200;

SS_x[25]=1.492*200;
SS_y[25]=(5-4.344)*200;

SS_x[26]=1.225*200;
SS_y[26]=(5-4.303)*200;

SS_x[27]=0.976*200;
SS_y[27]=(5-4.13)*200;

SS_x[28]=0.749*200;
SS_y[28]=(5-3.93)*200;

SS_x[29]=0.633*200;
SS_y[29]=(5-3.635)*200;
/*

SS_x[30]=3.14*200;
SS_y[30]=(5-0.87)*200;

SS_x[31]=2.586*200;
SS_y[31]=(5-0.876)*200;

SS_x[32]=2.455*200;
SS_y[32]=(5-0.5721)*200;

SS_x[33]=1.63*200;
SS_y[33]=(5-0.596)*200;

SS_x[34]=0.924*200;
SS_y[34]=(5-0.87)*200;

SS_x[35]=3.16*200;
SS_y[35]=(5-4.31)*200;

SS_x[36]=4.165*200;
SS_y[36]=(5-1.061)*200;

SS_x[37]=3.754*200;
SS_y[37]=(5-0.727)*200;

SS_x[38]=1.26*200;
SS_y[38]=(5-4.11)*200;

SS_x[39]=4.29*200;
SS_y[39]=(5-3.25)*200;
*/

 ////////////////////////////////////////////////////////
//print max,min
 fprintf(f1,"%s","max: ");

 for(ne=0;ne<no_of_RS;ne++)
 {
        for(br=0;br<no_of_SS;br++)
        {
                dis_x = (SS_x[br] - RS_x[ne])*(SS_x[br] - RS_x[ne]);
                dis_y = (SS_y[br] - RS_y[ne])*(SS_y[br] - RS_y[ne]);
                fprintf(f1," %lf %c %c%d%d%c%c%c",(1/sqrt(dis_x+dis_y)),'*','x',br,ne,' ','+',' ');
        }
 }



fseek(f1,-2,1);
fprintf(f1,"%c",';');
```

```
fprintf(f1,"%c%c",'\n','\n');

//print x00<1
for(ne=0;ne<no_of_RS;ne++)
 {
		for(br=0;br<no_of_SS;br++)
		{
			fprintf(f1,"%c%d%d %c%c%d%c%c",'x',br,ne,'<','=',1,';','\n');
		}
 }

fprintf(f1,"%c%c",'\n','\n');

//print x00>0
for(ne=0;ne<no_of_RS;ne++)
 {
		for(br=0;br<no_of_SS;br++)
		{
			fprintf(f1,"%c%d%d %c%c%d%c%c",'x',br,ne,'>','=',0,';','\n');
		}
 }

fprintf(f1,"%c%c",'\n','\n');


//x00+x01 <=1;
for(br=0;br<no_of_SS;br++)
 {
		for(ne=0;ne<no_of_RS;ne++)
		{
			fprintf(f1,"%c%d%d %c ",'x',br,ne,'+');
		}
		 fseek(f1,-2,1);
		fprintf(f1," %c%c  %d%c",'<','=',1,';');
		fprintf(f1,"%c",'\n');
 }

fprintf(f1,"%c%c",'\n','\n');

 for(i=0;i<no_of_RS;i++)
 {
		cc=0;
		for(j=0;j<no_of_SS;j++)
		{
			//flag1=0;

				Beam_sets[i][cc][temp]=j;
					temp++;

			for(k=(j+1);k<no_of_SS;k++)
			{

				// nu=k%no_of_SS;
				 nu=k;
			//	 printf(" %d %d %d \n",i,j,nu);
				 angle_frm_fun[a] = fun_angle(RS_x[i],RS_y[i],SS_x[j],SS_y[j],SS_x[nu],SS_y[nu]);
				printf(" abgle b/w RS[%d] ss[%d] ss[%d] = %f \n",i,j,nu,angle_frm_fun[a]);



					if(angle_frm_fun[a]<(float)40 )
					{
						printf("2 %d %d \n",j,nu);

						Beam_sets[i][cc][temp]=nu;
						temp++;
```

```
                                                                }


                                        }

                                        max_third[i][cc]=temp;
                                       temp=0;
                                        cc++;


                                }



                        // y=0;
                }

                printf(" \n\n");
                 for(i=0;i<no_of_RS;i++)
                {

                        for(j=0;j<no_of_SS;j++)
                        {
                                for(k=0;k<max_third[i][j];k++)
                                {
                                        printf("b[%d][%d] = %d    ",i,j,Beam_sets[i][j][k]);
                                }
                                printf("\n");
                        }
                 }

                 //ss<1;
                 for(i=0;i<no_of_RS;i++)
                {

                        for(j=0;j<no_of_SS;j++)
                        {
                                for(k=0;k<max_third[i][j];k++)
                                {
                                        if(Beam_sets[i][j][k] == j)
                                        {

                                                //              printf("%c%d%d %c ",'s',i,l,'+');
                                                                fprintf(f1,"%c%d%d %c%c%d%c%c ",'s',i,j,'<','=',1,';','\n');
                                        }
                                }
                        }

   }

                  fprintf(f1,"%c",'\n');
                fprintf(f1,"%c",'\n');


// ss00 + ss01 = 1;
//printf("\n\n");



   for(i=0;i<no_of_RS;i++)
                {

                        for(j=0;j<no_of_SS;j++)
                        {
                                for(k=0;k<max_third[i][j];k++)
```

```
                                {
                                        if(Beam_sets[i][j][k] == j)
                                        {

                                                //              printf("%c%d%d %c ",'s',i,l,'+');
                                                                fprintf(f1,"%c%d%d %c ",'s',i,j,'+');
                                        }
                                }
                        }
                        fseek(f1,-2,1);
                        fprintf(f1," %c %d%c",'=',1,';');
                        fprintf(f1,"%c",'\n');
  }

   fprintf(f1,"%c",'\n');
            fprintf(f1,"%c",'\n');

            //s00>=x10;

for(l=0;l<no_of_SS;l++)
{
   for(i=0;i<no_of_RS;i++)
            {

                        for(j=0;j<no_of_SS;j++)
                        {
                                for(k=0;k<max_third[i][j];k++)
                                {
                                        if(Beam_sets[i][j][k] == l)
                                        {

                                                                fprintf(f1,"%c%d%d %c ",'s',i,j,'+');
                                        }
                                }

                        }
                        fseek(f1,-2,1);
                        fprintf(f1," %c%c %c%d%d",'>','=','x',l,i);
                        fprintf(f1,"%c%c",';','\n');

  }

}









/*


            for(x=0;x<no_of_SS;x++)
            {
                        for(alp=0;alp<no_of_RS;alp++)
                        {
                                for(bet=0;bet<no_of_SS;bet++)
                                {
```

```
                                        if(Beam_sets[alp][bet]>-1 && Beam_sets[alp][bet]<1000)
                                        {
                                                // printf("%d",x);

                                                if(Beam_sets[alp][bet] <10 && Beam_sets[alp][bet] == x+1)
                                                {
                                                        S[x][alp]=Beam_sets[alp][bet];
                                                        printf(" S[%d][%d] = B[%d][%d] \n",x+1,alp, alp,bet);

                                                }

                                                if(Beam_sets[alp][bet] >9 && Beam_sets[alp][bet]/10 == x+1)
                                                {
                                                        S[x][alp]=Beam_sets[alp][bet];
                                                        printf(" S[%d][%d] = B[%d][%d] \n",x+1,alp, alp,bet);
                                                }
                                                else{

                                                        if(Beam_sets[alp][bet] >9 && Beam_sets[alp][bet]%10 == x+1)
                                                {
                                                        S[x][alp]=Beam_sets[alp][bet];
                                                        printf(" S[%d][%d] = B[%d][%d] \n",x+1,alp, alp,bet);
                                                }
                                                }


                                        }
                                }
                        }
                }


        for(rw=0;rw<no_of_RS;rw++)
        {
                for(nt=0;nt<no_of_SS;nt++)
                {
                        //if nxt num not contain this number
                        if(S[nt][rw]>9)
                        {
                                fprintf(f1,"%c%d%d %c%c %c%d%d%c%c",'s',rw,la,'>','=','x',S[nt][rw]/10,rw,';','\n');
                                fprintf(f1,"%c%d%d %c%c %c%d%d%c%c",'s',rw,la,'>','=','x',S[nt][rw]%10,rw,';','\n');
                        }
                        else
                        {
                                fprintf(f1,"%c%d%d %c%c %c%d%d%c%c",'s',rw,la,'>','=','x',S[nt][rw],rw,';','\n');
                        }

                        //if nxt num comtain this number
                }
        }


*/
```

```
//if(BS[aa][bb]=1)
 //printf(" B[%d][%d] has a angle of %f \n",aa,bb,Beam_sets[aa][bb]);

		//printf("sdf");

		 return 0;


}

float fun_angle(r_x,r_y,s1_x,s1_y,s2_x,s2_y)
float r_x,r_y,s1_x,s1_y,s2_x,s2_y;
{
		float final_angle[no_of_SS];
		int m=0;
		int isNan;
		float dis_b = (float)sqrt( ((s2_x-s1_x)*(s2_x-s1_x))+((s2_y-s1_y)*(s2_y-s1_y)) );
		float dis_a = (float)sqrt( ((s2_x-r_x)*(s2_x-r_x))+((s2_y-r_y)*(s2_y-r_y)) );
		float dis_c = (float)sqrt( ((s1_x-r_x)*(s1_x-r_x))+((s1_y-r_y)*(s1_y-r_y)) );

		float angle = (((dis_a*dis_a) + (dis_c*dis_c) - (dis_b*dis_b))/(2*dis_a*dis_c));

		//if(s2_x == (float)751)
		// printf("-------- %f \n", angle);

		if( (float)angle != (float)1.000000 )
		{

		final_angle[m] = (float)acos((float)angle)*(float)180.0;
		final_angle[m] = final_angle[m]/(float)3.14;



		isNan = (_isnan(final_angle[m])!=0);

		if(isNan!=0 && (int)angle>0)
			final_angle[m]=(float)0;

		if(isNan!=0 && (int)angle<0)
			final_angle[m]=(float)180;




		}
		if( (float)angle == (float)1.000000 )
		{
			if(s2_x == (float)751)
			 printf("a1= %f \n",angle);
			final_angle[m] = (float)0;


		}
		//if(s2_x == (float)751)
		 //printf("-------- %f \n", final_angle[m]);




		return(final_angle[m]);
		m++;
}
```

APPENDIX E

ROUNDED CODE FOR OPNET COORDINATES

```
// brendan_alg_5.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include<math.h>
#include <stdio.h>
#include <stdlib.h>

#define no_of_RS 7
#define no_of_SS 30

int main()
{
		float rate[no_of_SS];
		int a9[no_of_SS];
		int a5[no_of_SS][99];
		double r_x[no_of_RS][no_of_SS];
		double r_y[no_of_RS][no_of_SS];
		double sum=0;
		int l=0;
		int k=0;
		int la=0;
		int ne = 0;
		int br=0;
		int rw = 0;
		int nt=0;
		int i,j=0;
		int b;
		int nu=0;
		 int a=0;
		 int z=0;
		 int rr=0;
		 int aa=0,bb=0;
		 int cc=0;
		 int dd=0;
		 int alp=0;
		 int bet=0;
		 float dis_x=0;
		 float dis_y=0;
		 int gm=0;
		 int mac_cc=0;
		 int S[no_of_SS][no_of_RS];
		 int temp = 0;
		 int flag1[no_of_SS];
		 int a1[no_of_RS];
		 int a2[no_of_RS];

		 FILE *f1;


		float RS_x[no_of_RS];
		float RS_y[no_of_RS];
		float angle_frm_fun[no_of_SS];
```

```
int Beam_sets[no_of_RS][100][100];
int beamset_temp[no_of_RS][100][100];
int max_third[100][100];

float s_sets[no_of_SS];
float BS[no_of_RS][no_of_SS];

//int SS_num = no_of_SS-1;
float SS_x[no_of_SS];
float SS_y[no_of_SS];

double x[30][no_of_RS];
double y[no_of_RS][99];

float fun_angle();
/*

//int RS_num = no_of_RS-1;
int l=0;
int k=0;
int la=0;
int ne = 0;
int br=0;
int rw = 0;
int nt=0;
int i,j=0;
int b;
int nu=0;
 int a=0;
 int z=0,y=0,x;
 int rr=0;
 int aa=0,bb=0;
 int cc=0;
 int dd=0;
 int alp=0;
 int bet=0;
 float dis_x=0;
 float dis_y=0;
 int gm=0;
 int mac_cc=0;
 int S[no_of_SS][no_of_RS];
 int temp = 0;
 int flag1[no_of_SS];

 FILE *f1;


float RS_x[no_of_RS];
float RS_y[no_of_RS];
float angle_frm_fun[no_of_SS];

int Beam_sets[no_of_RS][100][100];
int beamset_temp[no_of_RS][100][100];
int max_third[100][100];

float s_sets[no_of_SS];
float BS[no_of_RS][no_of_SS];

//int SS_num = no_of_SS-1;
float SS_x[no_of_SS];
float SS_y[no_of_SS];

float fun_angle();
*/
//assignment of RS

f1=fopen("var.txt","w");
```

```
 RS_x[0]=2.056*200;
RS_y[0]=(5-2.88)*200;

RS_x[1]=1.92*200;
RS_y[1]=(5-2.41)*200;

RS_x[2]=2.096*200;
RS_y[2]=(5-1.986)*200;

RS_x[3]=2.526*200;
RS_y[3]=(5-1.882)*200;

RS_x[4]=2.445*200;
RS_y[4]=(5-2.973)*200;

RS_x[5]=2.875*200;
RS_y[5]=(5-2.95)*200;

RS_x[6]=3.24*200;
RS_y[6]=(5-2.84)*200;

// RS_x[1]=500;
// RS_y[1]=500;

//  RS_x[1]=100;
// RS_y[1]=120;

 //assifnment of ss
/*
 SS_x[0]=250;
 SS_y[0]=250;

 SS_x[1]=250;
 SS_y[1]=250;

 SS_x[2]=250;
 SS_y[2]=750;

 SS_x[3]=251;
 SS_y[3]=751;

 SS_x[4]=750;
 SS_y[4]=750;

 SS_x[5]=750;
 SS_y[5]=751;

 SS_x[6]=750;
 SS_y[6]=250;

 SS_x[7]=751;
 SS_y[7]=251;
 */

SS_x[0]=0.604*200;
SS_y[0]=(5-3.304)*200;

SS_x[1]=0.563*200;
SS_y[1]=(5-2.96)*200;

SS_x[2]=0.581*200;
SS_y[2]=(5-2.62)*200;

SS_x[3]=0.546*200;
SS_y[3]=(5-2.27)*200;
```

```
SS_x[4]=0.639*200;
SS_y[4]=(5-2.09)*200;

SS_x[5]=0.6911*200;
SS_y[5]=(5-1.835)*200;

SS_x[6]=0.685*200;
SS_y[6]=(5-1.5)*200;

SS_x[7]=0.883*200;
SS_y[7]=(5-1.35)*200;

SS_x[8]=1.07*200;
SS_y[8]=(5-1.18)*200;

SS_x[9]=1.3*200;
SS_y[9]=(5-1.1)*200;

SS_x[10]=1.55*200;
SS_y[10]=(5-0.952)*200;

SS_x[11]=1.81*200;
SS_y[11]=(5-0.877)*200;

SS_x[12]=2.12*200;
SS_y[12]=(5-0.842)*200;

SS_x[13]=2.387*200;
SS_y[13]=(5-0.7724)*200;

SS_x[14]=4.129*200;
SS_y[14]=(5-3.345)*200;

SS_x[15]=4.18*200;
SS_y[15]=(5-3.69)*200;

SS_x[16]=3.972*200;
SS_y[16]=(5-3.955)*200;

SS_x[17]=3.728*200;
SS_y[17]=(5-4.135)*200;

SS_x[18]=3.51*200;
SS_y[18]=(5-4.26)*200;

SS_x[19]=3.299*200;
SS_y[19]=(5-4.367)*200;

SS_x[20]=3.04*200;
SS_y[20]=(5-4.28)*200;

SS_x[21]=2.776*200;
SS_y[21]=(5-4.315)*200;

SS_x[22]=2.364*200;
SS_y[22]=(5-4.547)*200;

SS_x[23]=1.986*200;
SS_y[23]=(5-4.53)*200;

SS_x[24]=1.742*200;
SS_y[24]=(5-4.437)*200;

SS_x[25]=1.492*200;
SS_y[25]=(5-4.344)*200;

SS_x[26]=1.225*200;
```

```
SS_y[26]=(5-4.303)*200;

SS_x[27]=0.976*200;
SS_y[27]=(5-4.13)*200;

SS_x[28]=0.749*200;
SS_y[28]=(5-3.93)*200;

SS_x[29]=0.633*200;
SS_y[29]=(5-3.635)*200;
/*

SS_x[30]=3.14*200;
SS_y[30]=(5-0.87)*200;

SS_x[31]=2.586*200;
SS_y[31]=(5-0.876)*200;

SS_x[32]=2.455*200;
SS_y[32]=(5-0.5721)*200;

SS_x[33]=1.63*200;
SS_y[33]=(5-0.596)*200;

SS_x[34]=0.924*200;
SS_y[34]=(5-0.87)*200;

SS_x[35]=3.16*200;
SS_y[35]=(5-4.31)*200;

SS_x[36]=4.165*200;
SS_y[36]=(5-1.061)*200;

SS_x[37]=3.754*200;
SS_y[37]=(5-0.727)*200;

SS_x[38]=1.26*200;
SS_y[38]=(5-4.11)*200;

SS_x[39]=4.29*200;
SS_y[39]=(5-3.25)*200;
*/

 ////////////////////////////////////////////////////////
//print max,min
 fprintf(f1,"%s","max: ");

 for(ne=0;ne<no_of_RS;ne++)
 {
        for(br=0;br<no_of_SS;br++)
        {
                dis_x = (SS_x[br] - RS_x[ne])*(SS_x[br] - RS_x[ne]);
                dis_y = (SS_y[br] - RS_y[ne])*(SS_y[br] - RS_y[ne]);
                 r_x[ne][br]=dis_x;
                r_y[ne][br]=dis_y;
                fprintf(f1," %lf %c %c%d%d%c%c%c",(1/sqrt(dis_x+dis_y)),'*','x',br,ne,' ','+',' ');
        }
 }



fseek(f1,-2,1);
fprintf(f1,"%c",';');
fprintf(f1,"%c%c",'\n','\n');

//print x00<1
for(ne=0;ne<no_of_RS;ne++)
```

```
{
		for(br=0;br<no_of_SS;br++)
		{
			fprintf(f1,"%c%d%d %c%c%d%c%c",'x',br,ne,'<','=',1,';','\n');
		}
}

fprintf(f1,"%c%c",'\n','\n');

//print x00>0
for(ne=0;ne<no_of_RS;ne++)
{
		for(br=0;br<no_of_SS;br++)
		{
			fprintf(f1,"%c%d%d %c%c%d%c%c",'x',br,ne,'>','=',0,';','\n');
		}
}

fprintf(f1,"%c%c",'\n','\n');


//x00+x01 <=1;
for(br=0;br<no_of_SS;br++)
{
		for(ne=0;ne<no_of_RS;ne++)
		{
			fprintf(f1,"%c%d%d %c ",'x',br,ne,'+');
		}
		 fseek(f1,-2,1);
		fprintf(f1," %c%c  %d%c",'<','=',1,';');
		fprintf(f1,"%c",'\n');
}

fprintf(f1,"%c%c",'\n','\n');

 for(i=0;i<no_of_RS;i++)
{
		cc=0;
		for(j=0;j<no_of_SS;j++)
		{
			//flag1=0;

				Beam_sets[i][cc][temp]=j;
					temp++;

			for(k=(j+1);k<no_of_SS;k++)
			{

				// nu=k%no_of_SS;
				 nu=k;
			//		 printf(" %d %d %d \n",i,j,nu);
				 angle_frm_fun[a] = fun_angle(RS_x[i],RS_y[i],SS_x[j],SS_y[j],SS_x[nu],SS_y[nu]);
				printf(" abgle b/w RS[%d] ss[%d] ss[%d] = %f \n",i,j,nu,angle_frm_fun[a]);



					if(angle_frm_fun[a]<(float)40 )
					{
						printf("2 %d %d \n",j,nu);

						Beam_sets[i][cc][temp]=nu;
						temp++;
					}
```

```
                                        }

                                        max_third[i][cc]=temp;
                                       temp=0;
                                        cc++;


                                }



                        // y=0;
                }

                printf(" \n\n");
                 for(i=0;i<no_of_RS;i++)
                {

                        for(j=0;j<no_of_SS;j++)
                        {
                                for(k=0;k<max_third[i][j];k++)
                                {
                                        printf("b[%d][%d] = %d    ",i,j,Beam_sets[i][j][k]);
                                }
                                printf("\n");
                        }
                 }

                 //ss<1;
                 for(i=0;i<no_of_RS;i++)
                {

                        for(j=0;j<no_of_SS;j++)
                        {
                                for(k=0;k<max_third[i][j];k++)
                                {
                                        if(Beam_sets[i][j][k] == j)
                                        {

                                                //              printf("%c%d%d %c ",'s',i,l,'+');
                                                                fprintf(f1,"%c%d%d %c%c%d%c%c ",'s',i,j,'<','=',1,';','\n');
                                        }
                                }
                        }

  }

                  fprintf(f1,"%c",'\n');
                fprintf(f1,"%c",'\n');


// ss00 + ss01 = 1;
//printf("\n\n");



  for(i=0;i<no_of_RS;i++)
                {

                        for(j=0;j<no_of_SS;j++)
                        {
                                for(k=0;k<max_third[i][j];k++)
                                {
                                        if(Beam_sets[i][j][k] == j)
                                        {
```

```
                                                //              printf("%c%d%d %c ",'s',i,l,'+');
                                                                fprintf(f1,"%c%d%d %c ",'s',i,j,'+');
                                                        }
                                                }
                                        }
                                        fseek(f1,-2,1);
                                        fprintf(f1," %c %d%c",'=',1,';');
                                        fprintf(f1,"%c",'\n');
        }

        fprintf(f1,"%c",'\n');
                        fprintf(f1,"%c",'\n');

                        //s00>=x10;

for(l=0;l<no_of_SS;l++)
{
        for(i=0;i<no_of_RS;i++)
                        {

                                        for(j=0;j<no_of_SS;j++)
                                        {
                                                        for(k=0;k<max_third[i][j];k++)
                                                        {
                                                                        if(Beam_sets[i][j][k] == l)
                                                                        {

                                                                                        fprintf(f1,"%c%d%d %c ",'s',i,j,'+');
                                                                        }
                                                        }

                                        }
                                        fseek(f1,-2,1);
                                        fprintf(f1," %c%c %c%d%d",'>','=','x',l,i);
                                        fprintf(f1,"%c%c",';','\n');

        }

}
```

```
                x[0][0]=1;
x[1][0]=1;
x[2][0]=1;
x[3][0]=0;
x[4][0]=0;
x[5][0]=0;
x[6][0]=0;
x[7][0]=0;
x[8][0]=0;
x[9][0]=0;
x[10][0]=0;
x[11][0]=0;
x[12][0]=0;
```

```
x[13][0]=0;
x[14][0]=0;
x[15][0]=0;
x[16][0]=0;
x[17][0]=0;
x[18][0]=0;
x[19][0]=0;
x[20][0]=0;
x[21][0]=0;
x[22][0]=0;
x[23][0]=0;
x[24][0]=0;
x[25][0]=0;
x[26][0]=0;
x[27][0]=1;
x[28][0]=1;
x[29][0]=1;

/*
x[40][0]=0;
x[41][0]=0;
x[42][0]=0;
x[43][0]=0;
x[44][0]=0;
x[45][0]=0;
x[46][0]=0;
x[47][0]=0;
x[48][0]=0;
x[49][0]=0;
x[50][0]=0;
x[51][0]=0;
*/
x[0][1]=0;
x[1][1]=0;
x[2][1]=0;
x[3][1]=1;
x[4][1]=1;
x[5][1]=1;
x[6][1]=1;
x[7][1]=1;
x[8][1]=0;
x[9][1]=0;
x[10][1]=0;
x[11][1]=0;
x[12][1]=0;
x[13][1]=0;
x[14][1]=0;
x[15][1]=0;
x[16][1]=0;
x[17][1]=0;
x[18][1]=0;
x[19][1]=0;
x[20][1]=0;
x[21][1]=0;
x[22][1]=0;
x[23][1]=0;
x[24][1]=0;
x[25][1]=0;
x[26][1]=0;
x[27][1]=0;
x[28][1]=0;
x[29][1]=0;

/*
x[40][1]=0;
x[41][1]=0;
x[42][1]=0;
```

```
x[43][1]=0;
x[44][1]=0;
x[45][1]=0;
x[46][1]=0;
x[47][1]=0;
x[48][1]=0;
x[49][1]=0;
x[50][1]=0;
x[51][1]=0;
*/
x[0][2]=0;
x[1][2]=0.0;
x[2][2]=0.0;
x[3][2]=0.0;
x[4][2]=0.0;
x[5][2]=0.0;
x[6][2]=0.0;
x[7][2]=0.0;
x[8][2]=1;
x[9][2]=1;
x[10][2]=1;
x[11][2]=1;
x[12][2]=0.0;
x[13][2]=0.0;
x[14][2]=0.0;
x[15][2]=0.0;
x[16][2]=0.0;
x[17][2]=0.0;
x[18][2]=0.0;
x[19][2]=0.0;
x[20][2]=0;
x[21][2]=0;
x[22][2]=0;
x[23][2]=0;
x[24][2]=0;
x[25][2]=0;
x[26][2]=0;
x[27][2]=0;
x[28][2]=0;
x[29][2]=0;

/*
x[40][2]=0;
x[41][2]=0;
x[42][2]=0;
x[43][2]=0;
x[44][2]=0;
x[45][2]=0;
x[46][2]=0;
x[47][2]=0;
x[48][2]=0;
x[49][2]=0;
x[50][2]=0;
x[51][2]=0;
*/
x[0][3]=0;
x[1][3]=0;
x[2][3]=0;
x[3][3]=0;
x[4][3]=0;
x[5][3]=0;
x[6][3]=0;
x[7][3]=0;
x[8][3]=0;
x[9][3]=0;
x[10][3]=0;
x[11][3]=0;
```

```
x[12][3]=1;
x[13][3]=1;
x[14][3]=0;
x[15][3]=0;
x[16][3]=0;
x[17][3]=0;
x[18][3]=0;
x[19][3]=0;
x[20][3]=0;
x[21][3]=0;
x[22][3]=0;
x[23][3]=0;
x[24][3]=0;
x[25][3]=0;
x[26][3]=0;
x[27][3]=0.0;
x[28][3]=0.0;
x[29][3]=0.0;

/*
x[40][3]=1;
x[41][3]=1;
x[42][3]=1;
x[43][3]=1;
x[44][3]=1;
x[45][3]=1;
x[46][3]=1;
x[47][3]=1;
x[48][3]=1;
x[49][3]=1;
x[50][3]=1;
x[51][3]=1;
*/
x[0][4]=0;
x[1][4]=0;
x[2][4]=0;
x[3][4]=0;
x[4][4]=0;
x[5][4]=0;
x[6][4]=0;
x[7][4]=0.0;
x[8][4]=0.0;
x[9][4]=0.0;
x[10][4]=0.0;
x[11][4]=0.0;
x[12][4]=0;
x[13][4]=0;
x[14][4]=0;
x[15][4]=0;
x[16][4]=0;
x[17][4]=0;
x[18][4]=0;
x[19][4]=0;
x[20][4]=0;
x[21][4]=0;
x[22][4]=1;
x[23][4]=1;
x[24][4]=1;
x[25][4]=1;
x[26][4]=1;
x[27][4]=0;
x[28][4]=0;
x[29][4]=0;

/*
x[40][4]=0;
x[41][4]=0;
```

```
x[42][4]=0;
x[43][4]=0;
x[44][4]=0;
x[45][4]=0;
x[46][4]=0;
x[47][4]=0;
x[48][4]=0;
x[49][4]=0;
x[50][4]=0;
x[51][4]=0;
*/
x[0][5]=0;
x[1][5]=0;
x[2][5]=0;
x[3][5]=0;
x[4][5]=0;
x[5][5]=0;
x[6][5]=0;
x[7][5]=0;
x[8][5]=0;
x[9][5]=0;
x[10][5]=0;
x[11][5]=0;
x[12][5]=0;
x[13][5]=0;
x[14][5]=0;
x[15][5]=0;
x[16][5]=0;
x[17][5]=0;
x[18][5]=1;
x[19][5]=1;
x[20][5]=1;
x[21][5]=1;
x[22][5]=0;
x[23][5]=0;
x[24][5]=0;
x[25][5]=0;
x[26][5]=0;
x[27][5]=0;
x[28][5]=0;
x[29][5]=0;

/*
x[40][5]=0;
x[41][5]=0;
x[42][5]=0;
x[43][5]=0;
x[44][5]=0;
x[45][5]=0;
x[46][5]=0;
x[47][5]=0;
x[48][5]=0;
x[49][5]=0;
x[50][5]=0;
x[51][5]=0;
*/
x[0][6]=0;
x[1][6]=0.0;
x[2][6]=0.0;
x[3][6]=0.0;
x[4][6]=0.0;
x[5][6]=0.0;
x[6][6]=0;
x[7][6]=0;
x[8][6]=0;
x[9][6]=0;
x[10][6]=0;
```

```
x[11][6]=0;
x[12][6]=0;
x[13][6]=0;
x[14][6]=1;
x[15][6]=1;
x[16][6]=1;
x[17][6]=1;
x[18][6]=0;
x[19][6]=0;
x[20][6]=0;
x[21][6]=0;
x[22][6]=0;
x[23][6]=0;
x[24][6]=0;
x[25][6]=0;
x[26][6]=0;
x[27][6]=0;
x[28][6]=0;
x[29][6]=0;


for(l=0;l<no_of_RS;l++)
                        {
                                for(i=0;i<30;i++)
                                {
                                        printf("x[%d][%d] = %lf \n",i,l,x[i][l]);
                                }
                                printf("\n");
        }

sum=0;

 for(i=0;i<no_of_RS;i++)
         {

                 for(j=0;j<no_of_SS;j++)
                 {
                         for(k=0;k<max_third[i][j];k++)
                                 {
                                         sum = sum + (x[Beam_sets[i][j][k]][i])* (1/(sqrt(r_x[i][j]*r_x[i][j] +
r_y[i][j]*r_y[i][j])));

                                         //printf("b[%d][%d] = %d    ",i,j,Beam_sets[i][j][k]);
                         }

                         y[i][j]=sum;

                         printf("***** y[%d][%d]=%lf\n",i,j,sum);

                        sum=0;
                        }
                printf("\n\n");
 }



 sum=0;
//              d1=0;
//              omega1=0;


                        for(i=0;i<no_of_RS;i++)
         {
                         sum=0;
```

```
                    for(j=0;j<no_of_SS;j++)
                    {
                            if(y[i][j]>sum)
                            {
                                    sum = y[i][j];
                                    a1[i]=i;
                                    a2[i]=j;
                            }



                    }
                    printf(" y[%d][%d] is max with value %lf\n",a1[i],a2[i],sum);
                            printf(" => s'[%d][%d] = 1  because B[%d][%d] has the highest reward y\n\n\n
",a1[i],a2[i],a1[i],a2[i]);
                            sum=0;
            }


            for(i=0;i<no_of_SS;i++)
            {
                    a9[i]=0;
            }




                            for(i=0;i<no_of_RS;i++)
            {


                            for(k=0;k<max_third[a1[i]][a2[i]];k++)
                            {
                                    l=Beam_sets[a1[i]][a2[i]][k];
                            //      printf(" i = %d \n",l);
                                    a5[l][k] = i;
                            //      a5[22][3]=1;
                                    if(a9[l]==0)
                                    {
                                            rate[l] = (100/(sqrt(r_x[l][l]*r_x[l][l] + r_y[l][l]*r_y[l][l])));
                                    a9[l]=1;
                                    printf(" a[%d][%d] = %d ",l,k,i);
                                     printf("\n");
                                    }
                                    else
                                    {
                                            if((100/(sqrt(r_x[l][l]*r_x[l][l] + r_y[l][l]*r_y[l][l])))>rate[l])
                                            {
                                                    rate[l] = (100/(sqrt(r_x[l][l]*r_x[l][l] + r_y[l][l]*r_y[l][l])));
                                                    printf("err new  a[%d][%d] = %d ",l,k,i);
                                                    printf("\n");
                                            }

                                    }


                            }
                            }
```

```
		return 0;


}

float fun_angle(r_x,r_y,s1_x,s1_y,s2_x,s2_y)
float r_x,r_y,s1_x,s1_y,s2_x,s2_y;
{
		float final_angle[no_of_SS];
		int m=0;
		int isNan;
		float dis_b = (float)sqrt( ((s2_x-s1_x)*(s2_x-s1_x))+((s2_y-s1_y)*(s2_y-s1_y)) );
		float dis_a = (float)sqrt( ((s2_x-r_x)*(s2_x-r_x))+((s2_y-r_y)*(s2_y-r_y)) );
		float dis_c = (float)sqrt( ((s1_x-r_x)*(s1_x-r_x))+((s1_y-r_y)*(s1_y-r_y)) );

		float angle = (((dis_a*dis_a) + (dis_c*dis_c) - (dis_b*dis_b))/(2*dis_a*dis_c));

		//if(s2_x == (float)751)
		// printf("-------- %f \n", angle);

		if( (float)angle != (float)1.000000 )
		{

		final_angle[m] = (float)acos((float)angle)*(float)180.0;
		final_angle[m] = final_angle[m]/(float)3.14;



		isNan = (_isnan(final_angle[m])!=0);

		if(isNan!=0 && (int)angle>0)
			final_angle[m]=(float)0;

		if(isNan!=0 && (int)angle<0)
			final_angle[m]=(float)180;




		}
		if( (float)angle == (float)1.000000 )
		{
			if(s2_x == (float)751)
			 printf("a1= %f \n",angle);
			final_angle[m] = (float)0;


		}
		//if(s2_x == (float)751)
		 //printf("-------- %f \n", final_angle[m]);



		return(final_angle[m]);
		m++;
}
```

APPENDIX F

ILP FORMULATION FOR NP HARD COORDINATES

```
// brendan_5.3.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include<math.h>
#include <stdio.h>
#include <stdlib.h>
#define no_of_RS 8



int _tmain(int argc, _TCHAR* argv[])
{
		//0
		int rw1;
		int flag_extra[100];
		int beamset[8][100][100];
		int beamset_temp[8][100][100];
		int max_third[100][100];
		int countt=0;
		int couna[100];
		int coun=0;
		int omega=0;
		int fin=0;
		int post=0;
		int prnt=0;
 // char beamset_string[8];
  char a[1000];
  int t=0;
		int m_x=0;
		//0.1
		int max[8];
		int div_by;


		int ne,br,rw,nt;
		//1st
		int alp = 0;
		int bet = 0;
		//1.1
		int i[8];
		int j;
		int k=0;
		int b=0;
		int c=0;
		int d=0;
		int e=0;

		unsigned long long int zz=0;
		int f;

		FILE *f1;
```

```
f1=fopen("var.txt","w");


srand( rand() %10 +1 );
printf("the random numbers are \n");
//2nd
for(j=0;j<8;j++)
{
		//srand(rand(rand()));
		i[j]=rand() %10 +1;
		printf(" i[%d] = %d\n ",j+1,i[j]);

}

//3rd
//i[1]=9;
for(j=0;j<8;j++)
{
		//i[j]=rand() %10 +1;
		printf(" i[%d] = %d\n ",j+1,i[j]);

}

for(j=0;j<8;j++)
{
		k = k + i[j];
}
if(k%2 == 0)
{
		printf(" k is even, k = %d\n\n  ",k);
}
else
{
		j=j-1;
		k=k+1;
		printf(" k is odd, k+1= %d\n\n ",k);
		printf(" previous i[%d] is %d ",j+1,i[j]);
		i[j] = i[j] + 1;
		printf(" now i[%d] is %d ",j+1,i[j]);
}


//i[0]=1;
//i[1]=2;
//i[2]=4;
//i[3]=6;
//i[4]=8;
//i[5]=10;
//i[6]=12;
//i[7]=14;
k=57;
b=k+2;
//b=59;
printf(" k+2 relays are %d \n\n",b);

//---------------------------------------------

//4th
for(d=0;d<8;d++)
{

		c=k/2 -i[d];

		//max is the 2nd max
		//max-3rd is the max for 3rd

		max[alp]=c+1;
```

```
		alp++;


		for(j=0;j<c;j++)
		{


			for(e=j;e<=j+i[d]-1;e++)
			{
				beamset[d][m_x][omega]=e;
				omega++;

			}


			//printf("+++++ %d\n",omega);

			max_third[d][countt]=omega;
			countt++;

			m_x++;
			omega=0;


		}
		f=k/2-i[d];
		//if(d==1)
			//printf("%d---\n",omega);

		for(j=f;j<f+i[d]+1;j++)
		{
			beamset[d][m_x][omega]=j;
					omega++;

		}
		//if(d==1)
		//	printf("%d---\n",j);



		m_x=0;

		max_third[d][countt]=omega;

			countt++;
			omega=0;
			countt=0;

}


alp=0;
//5th

for(d=0;d<8;d++)
{

		c=k/2 -i[d];

		//max is the 2nd max
		//max-3rd is the max for 3rd

		//printf("*******%d\n",max[alp]);
		m_x=max[alp];
		alp++;
```

```
                        for(j=0;j<c;j++)
                        {


                                        for(e=j;e<=j+i[d]-1;e++)
                                {
                                        beamset[d][m_x][omega]=e+k/2+1;
                                        omega++;

                                }



                                m_x++;
                                omega=0;


                        }
                        f=k/2-i[d];

                        for(j=f;j<f+i[d]+1;j++)
                        {
                                beamset[d][m_x][omega]=j+k/2+1;
                                        omega++;

                        }



                        m_x=0;

                                        omega=0;

                }

printf("------------------------\n\n\n\n");

//6
                for(d=0;d<8;d++)
                {
                        for(bet=0;bet<max[d];bet++)
                        {
                                for(omega=0;omega<max_third[d][bet];omega++)
                                {

                //              beamset_temp[d][bet]=beamset[d][bet];
                                //beamset_string[d][bet]=atoi(beamset[d][bet]);
                                        printf("b[%d][%d] = %d    ",d,bet,beamset[d][bet][omega]);
                                                //printf("b[%d][%d] = %llu    ",d,bet,beamset[d][0][0]);
                                        }
                                        prnt++;
                                        printf("\n");

                        }
                        for(bet=0;bet<max[d];bet++)
                        {
                                for(omega=0;omega<max_third[d][bet];omega++)
                                {

                //              beamset_temp[d][bet]=beamset[d][bet];
                                //beamset_string[d][bet]=atoi(beamset[d][bet]);
                                        printf("b[%d][%d] = %d    ",d,bet+max[d],beamset[d][bet+max[d]][omega]);
                                                //printf("b[%d][%d] = %llu    ",d,bet,beamset[d][0][0]);
```

```
                                        }
                                        prnt++;
                                        printf("\n");

                                }
                                printf("\n\n");
                        }



////////////////////////////////////////////////////////////////////////////
///////////////merger/////////////////////////////////////////

 //print max,min
                        fprintf(f1,"%s","max: ");

                        for(ne=0;ne<no_of_RS;ne++)
                        {
                                for(br=0;br<k+2;br++)
                                {
                                        fprintf(f1,"%c%d%d%c%c%c",'x',br,ne,' ','+',' ');
                                }
                        }


//print x00<1
                        fseek(f1,-2,1);
                        fprintf(f1,"%c",';');
                        fprintf(f1,"%c%c",'\n','\n');

                        for(ne=0;ne<no_of_RS;ne++)
                        {
                                for(br=0;br<k+2;br++)
                                {
                                        fprintf(f1,"%c%d%d %c%c%d%c%c",'x',br,ne,'<','=',1,';','\n');
                                }
                        }

                        fprintf(f1,"%c%c",'\n','\n');

                        //print x00+x01<1
                        for(ne=0;ne<k+2;ne++)
                        {
                                for(br=0;br<no_of_RS;br++)
                                {
                                        fprintf(f1,"%c%d%d %c ",'x',ne,br,'+');
                                }
                                fseek(f1,-2,1);
                                fprintf(f1,"%c%c%d%c%c",'<','=',1,';','\n');


                        }

                        fprintf(f1,"%c%c",'\n','\n');

alp=0;
//print ss<1

//print ss<1

                                                for(br=0;br<k+2;br++)
                                                {
                                                        for(rw=0;rw<no_of_RS;rw++)
                                                        {
                                                                for(nt=0;nt<(max[rw]);nt++)
                                                                {
```

```
                                                for(omega=0;omega<max_third[rw][nt];omega++)
                                                {
                                                        if(beamset[rw][nt][omega]==br)
                                                                fprintf(f1,"%c%d%d
%c%c%d%c%c",'s',rw,nt,'<','=',1,';','\n');

                                                }


                                        }
                                        for(nt=0;nt<(max[rw]);nt++)
                                        {
                                                for(omega=0;omega<max_third[rw][nt];omega++)
                                                {
                                                        if(beamset[rw][nt+max[rw]][omega]==br)
                                                                fprintf(f1,"%c%d%d
%c%c%d%c%c",'s',rw,nt+max[rw],'<','=',1,';','\n');

                                                }


                                        }

                                }
                                fprintf(f1,"%c%c",'\n','\n');
                        }




 fprintf(f1,"%c%c",'\n','\n');
 fprintf(f1,"%c%c",'\n','\n');



 //ss+ss=1
//fin=0;

                for(rw=0;rw<no_of_RS;rw++)
                                        {
                                 for(rw1=0;rw1<99;rw1++)
                                        {
                                                flag_extra[rw1]=0;
                                        }
                                  for(br=0;br<k+2;br++)
                                {

                                        for(nt=0;nt<(max[rw]);nt++)
                                        {
                                                for(omega=0;omega<max_third[rw][nt];omega++)
                                                {
                                                        if(beamset[rw][nt][omega]==br)
                                                        {
                                                                if(flag_extra[nt]==0)
                                                                {
                                                                        flag_extra[nt]=1;
                                                                        fprintf(f1,"%c%d%d
%c ",'s',rw,nt,'+');

                                                                }
                                                        }

                                                }
```

```
                                                }
                                                for(nt=0;nt<(max[rw]);nt++)
                                                {
                                                        for(omega=0;omega<max_third[rw][nt];omega++)
                                                        {
                                                                if(beamset[rw][nt+max[rw]][omega]==br)
                                                                        if(flag_extra[nt+max[rw]]==0)
                                                                        {

        flag_extra[nt+max[rw]]=1;
                                                                                fprintf(f1,"%c%d%d
%c ",'s',rw,nt+max[rw],'+');
                                                                        }



                                                        }


                                                }



                                        }

                                        //fprintf(f1,"%c%c",'\n','\n');
                                          fseek(f1,-2,1);
                        fprintf(f1," %c%d",'=',1);
                        fprintf(f1,"%c%c",';','\n');
                        fprintf(f1,"%c%c",'\n','\n');
                                                }


fprintf(f1,"%c%c",'\n','\n');
fprintf(f1,"%c%c",'\n','\n');




//ss+ss>=x

 for(br=0;br<k+2;br++)
                                                {
                                        for(rw=0;rw<no_of_RS;rw++)
                                        {
                                                for(nt=0;nt<(max[rw]);nt++)
                                                {
                                                        for(omega=0;omega<max_third[rw][nt];omega++)
                                                        {
                                                                if(beamset[rw][nt][omega]==br)
                                                                        fprintf(f1,"%c%d%d %c
','s',rw,nt,'+');


                                                        }


                                                }
                                                for(nt=0;nt<(max[rw]);nt++)
                                                {
                                                        for(omega=0;omega<max_third[rw][nt];omega++)
                                                        {
                                                                if(beamset[rw][nt+max[rw]][omega]==br)
```

```
                                                                                fprintf(f1,"%c%d%d %c
",'s',rw,nt+max[rw],'+');

                                                                                }

                                                                        }

                                                                        fseek(f1,-2,1);
                                        fprintf(f1,"%c%c %c%d%d%c",'>','=','x',br,rw,';');
                                        fprintf(f1,"%c",'\n');

                                                                }

                                                                fprintf(f1,"%c%c",'\n','\n');
                                                }




 fprintf(f1,"%c%c",'\n','\n');
 fprintf(f1,"%c%c",'\n','\n');



fprintf(f1,"%c%c%c  ",'i','n','t');
for(ne=0;ne<no_of_RS;ne++)
        {
                for(br=0;br<k+2;br++)
                {
                        fprintf(f1,"%c%d%d%c   ",'x',br,ne,',');
                }
        }
fseek(f1,-4,1);
                fprintf(f1,"%c",';');
        fprintf(f1,"%c%c",'\n','\n');


        fprintf(f1,"%c%c%c  ",'i','n','t');
 for(br=0;br<k+2;br++)
                                                {
                                                        for(rw=0;rw<no_of_RS;rw++)
                                                        {
                                                                for(nt=0;nt<(max[rw]);nt++)
                                                                {
                                                                        for(omega=0;omega<max_third[rw][nt];omega++)
                                                                        {
                                                                                if(beamset[rw][nt][omega]==br)
                                                                                        fprintf(f1,"%c%d%d %c
",'s',rw,nt,',');

                                                                        }


                                                                }
                                                                for(nt=0;nt<(max[rw]);nt++)
                                                                {
                                                                        for(omega=0;omega<max_third[rw][nt];omega++)
                                                                        {
                                                                                if(beamset[rw][nt+max[rw]][omega]==br)
```

```
                                                                    fprintf(f1,"%c%d%d %c
","s",rw,nt+max[rw],',');

                                                                }


                                                            }

                                                        }
                                                    // fprintf(f1,"%c%c",'\n','\n');
                                                }
fseek(f1,-4,1);
                        fprintf(f1,"%c",';');
            fprintf(f1,"%c%c",'\n','\n');


            return 0;
}
```