**ENTS 656 Introduction to Cellular Communications**

**Spring 2016 Project**

**Due By Email: May 10, 5:00pm**

Write a Python application which will simulate the downlink behavior of a 3-sectored basestation along a road. The basestation is oriented so that one sector (the alpha sector) points north and the other two sectors point southeast, 120° from north, and southwest, 240° from north (the beta and gamma sectors respectively). The road runs north-south and is located east of the basestation. Since the gamma sector will not be involved with covering the road, you should simulate the alpha and beta sectors ONLY. The following picture should help illustrate this:

Road

α

β

basestation

γ

This picture uses a standard 3-arrow symbol for the basestation where each arrow shows where the antennas are pointing. In terms of a hexagonal cell layout, such a basestation located at the center of a hexagon would cover the top, lower right and lower left parts of the hexagon as illustrated below (in black and white and also in color):

For realism, I’ve included a dotted line in the picture with the roads that shows what the actual coverage area of such a basestation might look like, though you will not need this for the project.

Your simulation will have the following basic parameters, some of which may be varied:

* **Road length**: 6 km or more
* **Simulation Time Step Size**: ΔT = 1 second
* **Total Simulation Time:** 1 or more hours.

Each sector of the basestation will have the following properties, some of which may be varied:

* **Height**: hB = 50 m (This is the basestation antenna height)
* **Location**: 20 m west of the road at the midpoint of the road (e.g. at 3 km if the road is 6 km)
* **TX Power:** PTX= 43 dBm
* **Line/connector Losses:** L= 2 db
* **Antenna Gain:** AGTX= 15 dBi (at boresight)
* **Number of Traffic Channels:** NCH = 15 per sector
* **Frequencies:**
	+ Alpha (0°): fMHZ = 860 MHz
	+ Beta: (120°) fMHZ = 865 MHz

The mobiles will have the following properties, some of which may be varied:

* **Mobile Height:** hm = 1.5 meter
* **Handoff margin**: HOm = 3 dB (this will be used to reduce ping-ponging between sectors)
* **Mobile Rx Threshold**: RSLTHRESH = -102 dBm

We will assume all of the users for this basestation are on the road within the prescribed length. Users will appear along the road when they try to make a call. Assume the users are uniformly distributed along the road with the following characteristics, some of which may be varied:

* **Number of Users:** U = 160
* **Call Rate: λ =**2 calls per hour (on average)
* **Average Call Duration:** H=3 minutes/call (= 180 seconds/call)
* **User Speed:** v=15 m/s ( = 54 kph = 33.553977 mph)
* **Direction:** Users appearing on the road have a 50/50 chance of heading north or south.

Your simulation should run with discrete time steps equal to one second each. At each second you will need to determine if each user, who does not currently have an active call, is going to make a call; this can be derived from the call rate, assuming users may make a call at any time with the same probability. (NOTE: we will assume our users are good citizens who are using hands-free devices while they are driving; therefore it would be impossible for them to have more than one call active at a time, or to make conference calls, texts, etc.) When the call is established, your program should determine the length of the call in seconds. Call lengths will be exponentially distributed with mean equal to the average call duration (this will be the scale factor needed by the exponential routine in numpy.random). Users are assumed to establish a call on the sector which appears with the highest RSL when the call is initiated. The RSL will be determined by computing the EIRP in the direction of the user from each sector and subtracting the path loss obtained by modelling the communications channel for that sector.

The channel properties will determine the total path loss, which will include the propagation loss, shadowing, and fading as follows:

* **Propagation Loss**: Use the Okamura-Hata model adjusted for a small city. Include the mobile height term as required by the model. Note that you will need to compute the distance from the mobile to the basestation to get the propagation loss.
* **Shadowing**: Assume a log-normal distribution (i.e. a normal distribution in dB) with a zero mean and a standard deviation of σ = 2 dB. (For this problem with the basestation so near the roads, using such a low value of σ is reasonable. For a more general location around the city, we might expect a higher value for σ, e.g. 5 db or so…)
	+ **NOTE**: Shadowing is location dependent, NOT time dependent. Thus your simulation should construct shadowing values for each 10m section of the roads and save these for use with all mobiles, depending on the mobile location on the roads.
* **Fading:** Use a Rayleigh distribution: the magnitude of a complex Gaussian distribution such that the real and complex parts have zero mean and unit variance.
	+ Your simulation will be computing RSL values by subtracting the results of propagation from the EIRP and adding the shadowing and fading results. In your program, this will be compared to the mobile threshold value. It is unreasonable to design a system which drops a call as soon as the RSL falls below the threshold. Most real systems can survive an isolated deep fade. To model this in your simulation, each time you sample the Rayleigh distribution, generate 10 samples, throw away the deepest fade value and report the second deepest value as the fade value. In this way, the simulated system will only drop the call if 2 out of 10 of the values would end up below threshold. (NB: fading is a short time effect: you should assume the same values for propagation and shadowing will apply for all 10 fading samples.)
	+ Note that the Rayleigh distribution described here (and also in the fourth Python homework assignment) gives linear values for the fading multiplier. You will need to convert to dB before **subtracting from** the other components to get a path loss value.

Once you compute the total path loss, you must subtract this from the EIRP in the direction of the mobile to get the RSL. However, the EIRP in the direction of the mobile will NOT simply be the EIRP of the basestation, since the mobile will usually not be directly in front of the antenna. To find the EIRP in the direction of the mobile, first compute the EIRP at boresight (directly in front of the antenna) and then subtract off the antenna discrimination based on the angle off of boresight where the mobile is located. The antenna discrimination values are given in dB and supplied by the manufacturer for each antenna model. The discrimination values for the antenna you will use will be available to you electronically. Note that you will need to compute the angle off of boresight to use these values. One easy way to do this is to compute Δx and Δy between the basestation and the mobile and then compute the angle between the vector (Δx, Δy) and a unit vector in the direction the antenna is pointing (e.g. (0,1) and (√3/2,-1/2) for alpha and beta, respectively). Note: the angle between 2 vectors *u* and *v* is given by:
$$θ=arccos\frac{v∙u}{\left‖v\right‖\left‖u\right‖}$$

As the mobiles travel it is likely that, if they have an active call up, they will need to handoff from one sector to another. At each time step your simulation will need to evaluate the RSL from each of the sectors. Note that although the propagation loss and shadowing will be the same for all sectors, the EIRP in the direction of the mobile AND the fading will need to be computed for each sector.

Your simulation should keep track of each call attempt and all calls. It should keep a table of active calls to update as the simulation time progresses. It should also keep track of various conditions that may arise such as hand off attempts, successful hand offs, hand off failures, blocked calls, dropped calls due to signal strength, dropped calls due to capacity, completed calls,etc. It should also keep track of the number of active calls on each sector (which you will need to know to determine if a call attempt blocks). Initialize the simulation by assuming that none of the users have active calls. Your simulation should calculate various initial settings that do not need to be calculated at every step such as the EIRP at boresight for each sector and the shadowing values for each position along the 2 roads (in 10m increments). The remaining steps are executed for every time step, ΔT, for the duration of the simulation time, Ttotal. Note that for easier understanding, the instructions for what to do if a user does not have a call up are listed first. When your program executes, however, it is important to first service the users who have active calls up before dealing with those who do not. This guarantees that requests for handoffs between sectors will take precedence over new call requests, which is as it should be. It also guarantees that if user 1 terminates a call, user 2 may claim the newly freed channel as a free channel (which may be a little optimistic, but that’s ok).

1. For each user that does not have a call up:
	1. Determine if the user makes a call request. Each user without an active call will have a probability = λ\*ΔT of making a call during that time step (NOTE: make sure the units match!!).
	2. If the user does not make a call request, we are done: go on to the next user.
	3. If the user DOES make a call request:
		1. Determine the user’s location along the road. Users are assumed to be uniformly distributed along the length of road (anywhere from 0 to 6 km for a 6 km road). The user’s location should be selected using floating point arithmetic.
		2. Determine the user’s direction (i.e. north or south).
		3. Find the RSL at the mobile from each sector. This will require the distance between the mobile and the basestation (for the Okamura-Hata model) as well as the angle off boresight (for the antenna discrimination). Pick the sector with the highest RSL to be the one that will service the call request. This will be called the serving sector.
		4. Determine if the RSL from the serving sector, RSLSERVER is greater than or equal to the RSL threshold, RSLTHRESH for the mobile. If not, that is, if RSLSERVER < RSLTHRESH then the call attempt fails and this is recorded as a dropped call due to signal strength for that sector. We are done with user and can go on to the next one.
		5. If the RSLSERVER ≥ RSLTHRESH then we can attempt to establish the call. Your program should check to see if there is an available channel on the basestation. If not, then the call is blocked; this is recorded as a blocked call due to capacity for this sector. At this point you should see if the other sector has sufficient signal strength to be the serving sector.
			1. If the second best sector has RSLSERVER ≥ RSLTHRESH, then check if a channel is available on that sector. If so, proceed to then establish the call on that sector as below
			2. If the call cannot be established on the other sector, then record this as a dropped call due to capacity for THE ORIGINAL SECTOR ONLY. We don’t want to penalize the other sector for trying to help out.
		6. If RSLSERVER ≥ RSLTHRESH and there is an available channel then the call is established. Your simulation should make an entry for this user in the table of active calls. Your program will need to determine the length of the call so it can keep a counter of the time left. As explained above, call lengths will be exponentially distributed. You will need to keep track of the user’s location, direction of motion, serving sector, serving RSL, call time left, etc. This entry will be updated at each future time step until the call completes (either successfully or unsuccessfully as described below)
2. For each user who DOES have a call up:
	1. Update the user’s location. We assume users will continue in the same direction for the duration of their call.
	2. Update the call time. If the user’s call timer has run out, then the call completes normally. Record this as a successful call on that sector. You can delete the entry in the active call list (or move it to an archive list). This will also free a channel on the serving sector. We are done with this user for now.
	3. Check if the user’s location has moved beyond the ends of the road. If so, then we assume the user has handed off to another basestation further down the road. Record this as a successful call on this sector (since we are not modeling the handoff procedure to other basestations, we will not record this as a successful handoff). This will also free a channel on the serving sector. We are done with this user for now.
	4. Calculate the RSLSERVER. This will be at a new location, with new EIRP, path loss, shadowing and fading values. If RSLSERVER < RSLTHRESH the call drops. Record this as a dropped call due to signal strength for the serving sector. The entry should be deleted from the active call list (or archived). The channel is now free and available for other users. We are done with this user for now.
	5. If RSLSERVER ≥ RSLTHRESH, compute the RSL for the other sector, RSLOTHER. If RSLOTHER > RSLSERVER +HOm, then there is a potential handoff. NOTE: HOm >0 forces the call to stay on a weaker sector a little longer, which limits excessive hand offs due to fading.
		1. Record this as a hand off attempt for this sector
		2. If the other sector has a channel available, then there will be a handoff. This sector becomes the new serving sector, so the active call entry for this call will need to be updated. This is recorded as a successful handoff out of the old serving sector. The call timer is not reset by the handoff and will continue to count in the next time step. A channel is freed on the old server and a channel is now in use on the new server for this call.
		3. If the other sector does NOT have a channel available, then there will be a handoff failure. This is recorded as a failed handoff out of the old sector. The call will NOT drop: it simply continues on the old sector in the next time step.
3. Once all users have been processed, your simulation should collect statistics and update information about the sectors. Some statistics are easier to compute while users are being processed (like numbers of dropped calls for various reasons, and channel occupancy) while others will be easier to compute after all the users are updated. After each hour of simulation time, your program should produce a report for the basestation. For each sector your program should give
	1. the number of channels currently in use
	2. the number of call attempts
	3. the number of successful calls
	4. the number of successful handoffs
	5. the number of handoff failures into and out of each sector
	6. the number of call drops due to low signal strength and also due to capacity
	7. the number of blocks due to capacity
	8. any other information you think will be useful

Your program should also produce a summary report at the end for the entire simulation time

Q1: Run your simulation for six hours with the parameters given. How many problems (e.g. drops, blocks, hand off failures) occur? What percentage of call attempts have a problem? Does this basestation perform well?

Q2: Change the length of the road from 6 km to 8 km and move the basestation so it is still at the midway point (20m west and 4 km up from the bottom). Rerun your simulation for six hours. How many problems occur now, and what is the new percentage? Compared to Q1, what is the main cause of the additional problems? (e.g. drops from signal strength? Or blocks from capacity? Or Hand off failures? Etc.)

Q3: Change the length of the road back to 6 km (and put the basestation back where it was). Double the number of users (from 160 to 320). Rerun your simulation for six hours. How many problems occur now, and what is the new percentage? Compared to Q1, what is the main cause of the additional problems (e.g. drops from signal strength? Or blocks from capacity? Or Hand off failures? Etc.)

Q4: Add a call to the numpy.random.seed function at the beginning of your code so it generates the same “random” numbers every time. Change the handoff margin (HOm) to 5 db and rerun, then change the hand off margin to 0 db and rerun (keep the users at 320 as in Q3 along with the rest of the simulation). What is the effect on the number of hand offs?

While the structure of the simulation is for you to decide/develop, I would recommend creating a module of functions that perform various pieces of the problem (e.g. path loss calculations, fading, EIRP calculations, etc.) and also a script to run the main simulation. The independent functions can be debugged and tested separately, which will make writing and debugging the main scripts easier.

Please submit the Python source code implementing the simulator and scripts to run it. In addition, write and submit a short report giving the results generated by the code and the answers to the questions given in Q1, Q2, Q3, and Q4. All material should be submitted ELECTRONICALLY as email attachments to mdellomo@umd.edu.

PLEASE NOTE: in addition to correctness and functionality, the code will also be evaluated for style. Thus, you should pay attention to software design and engineering issues such as: code modularization, code block organization, variable naming, comments, etc.

A word about academic integrity: Copying code (or rewriting code or copying algorithms or sharing code etc.) from other students is strictly prohibited and will not be tolerated. You may discuss the project ONLY IN GENERAL TERMS with other students. If you are having problems, please see either the TA or me during our office hours. Please submit a signed honor pledge form with your code (see below)

HONOR PLEDGE: I pledge on my honor that I have not given or received any unauthorized assistance on this assignment.

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_