April, 1997

# UTMOST: A Traffic Modeling Tool for Wireless Network Design

By: C. Edward Chow and Heidi McClure

EAS-CS-97-1

University of Colorado at Colorado Springs
Department of Computer Science
College of Engineering and Applied Science

1420 Austin Bluffs Parkway
P.O. Box 7150
Colorado Springs, CO, 80933-7150

# UTMOST: A Traffic Modeling Tool for Wireless Network Design

**C. Edward Chow and Heidi McClure**

**Department of Computer Science**
**University of Colorado at Colorado Springs**
**Colorado Springs, CO 80933-7150**
**Tel: (719)262-3110**
**FAX: (719)262-3360**
**Email: {chow, mhmcclur}@quandary.uccs.edu**
**WWW: http://www.cs.uccs.edu/{~chow, ~mhmcclur}**

**Abstract**

**In this paper we present the design and implementation of a user traffic modeling and simulation tool for wireless network design. It has a graphical user interface for displaying the cell layout, user traversing patterns, the current power and cell site assignments. For each simulation step, it generates an output file of the current location of the users, their power levels and cell assignments, and the boundary crossing times and locations. It also computes the visible sectors for each mobile user. It can interact with dynamic channel assignment optimization packages and facilitate the verification of the assignment results. It outputs event traces to drive discrete event simulators for the cell size determination and the comparison of efficient hand-off algorithms.**

## 1. Introduction

The Colorado Advanced Software Institute (CASI) in conjunction with US West has provided us a grant to work on Resource Allocation and Admission Control Evaluator for Wireless Information Networks (RACEWIN). The initial effort is to provide a User Traffic MOdeling and Simulation Tool, called UTMOST, for the design of Personal Communications Services (PCS) Wireless Information Networks [1,2,3,4,5,6,7,8,9]. This paper is about the design and implementation of UTMOST and the important design issues encountered.

One of the design goals of UTMOST is to have a graphical user interface for visualizing the cell layout, the traveling patterns of mobile stations, and the power and cell site assignments of mobile stations. It enables a designer to verify the power and cell site assignment results generated by dynamic channel assignment procedures. For portability, we have chosen Java as the implementation language. UTMOST was implemented using the Abstract Window Toolkit (AWT) package in Java Development Kit (JDK) 1.0.2 [10,11, 12].

The other goal of UTMOST is to generate wireless network data as input for dynamic channel assignment algorithms, power and cell site assignment packages, and wireless network discrete event simulators , which simulates the wireless network resources for determining cell size and quality of service, and for designing efficient hand-off algorithms. The data generated by UTMOST include the user location, the traveling direction and speed, the current power level , the current sector and sector antennae assignment. For the optimization package, it also computes the set of visible sector antennae. For the discrete event simulators, it computes the times and places of boundary crossing events and generates the events of power on/off, roaming, or calling. The algorithms for computing these information are discussed in this paper.

To interface with other packages, the input and output data formats are defined to allow UTMOST and other wireless network design packages to communicate either interactively via sockets or in batch operating mode via file read/write.

## 2. Graphical User Interface of UTMOST

Figure 1 show the screen layout of UTMOST. It shows users as colored dots on a cell layout map of sectors. A PCS cell is represented by a hexagon. A sector is one third of a PCS cell. Each hexagon has at its center a base station. This base station has actually three antennae which are directionally aimed. Each sector represents the area "owned" by one of these directional antennae.

Each cell has a radius. Currently, in UTMOST, the radius is five (5) kilometers. It can be changed via a configuration file. On the maps, the radius is the same as one of the lines radiating out from a base station (center of hexagon) to the edge of that hexagon.

### 2.1 The traffic patterns

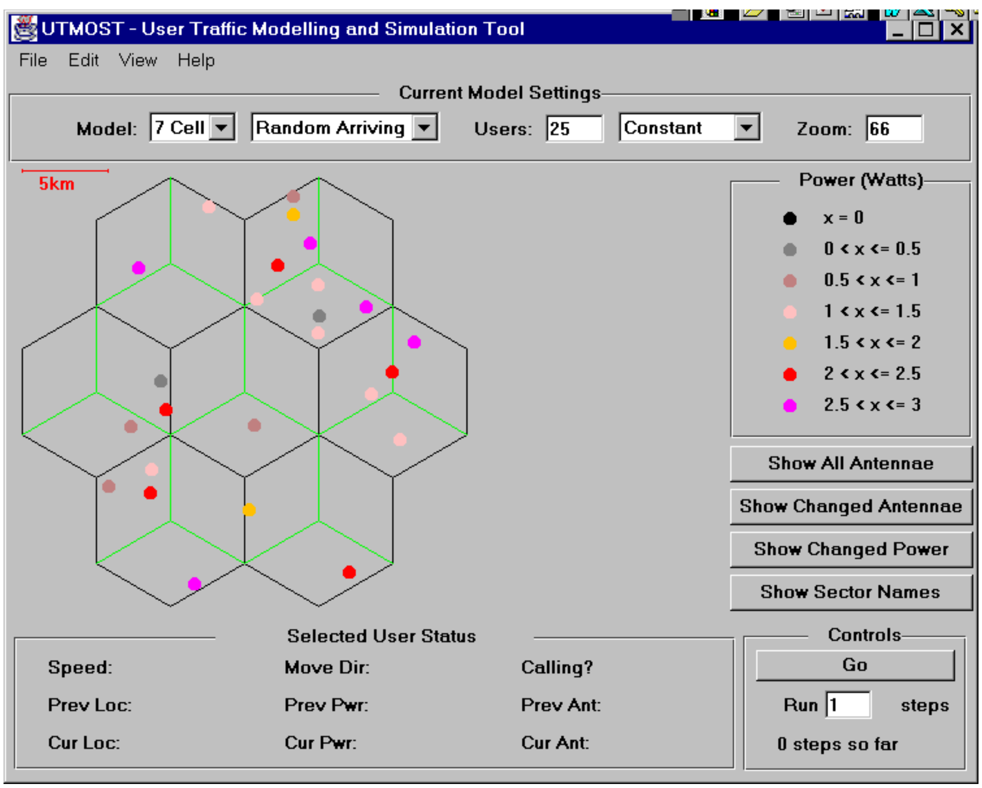


Figure 1. The Screen Layout of UTMOST.

UTMOST generates data for various traffic patterns. From one (1) to five thousand (5000) mobile stations may be modeled. Currently it includes a seven cell circular layout pattern and a four cell linear layout patterns. The seven cell circular model simulates the events where mobile users gather at the central cell (e.g. a football stadium). A four cell linear model simulates highway traffic.

Arriving, leaving and random patterns may be selected. The users may start randomly inside

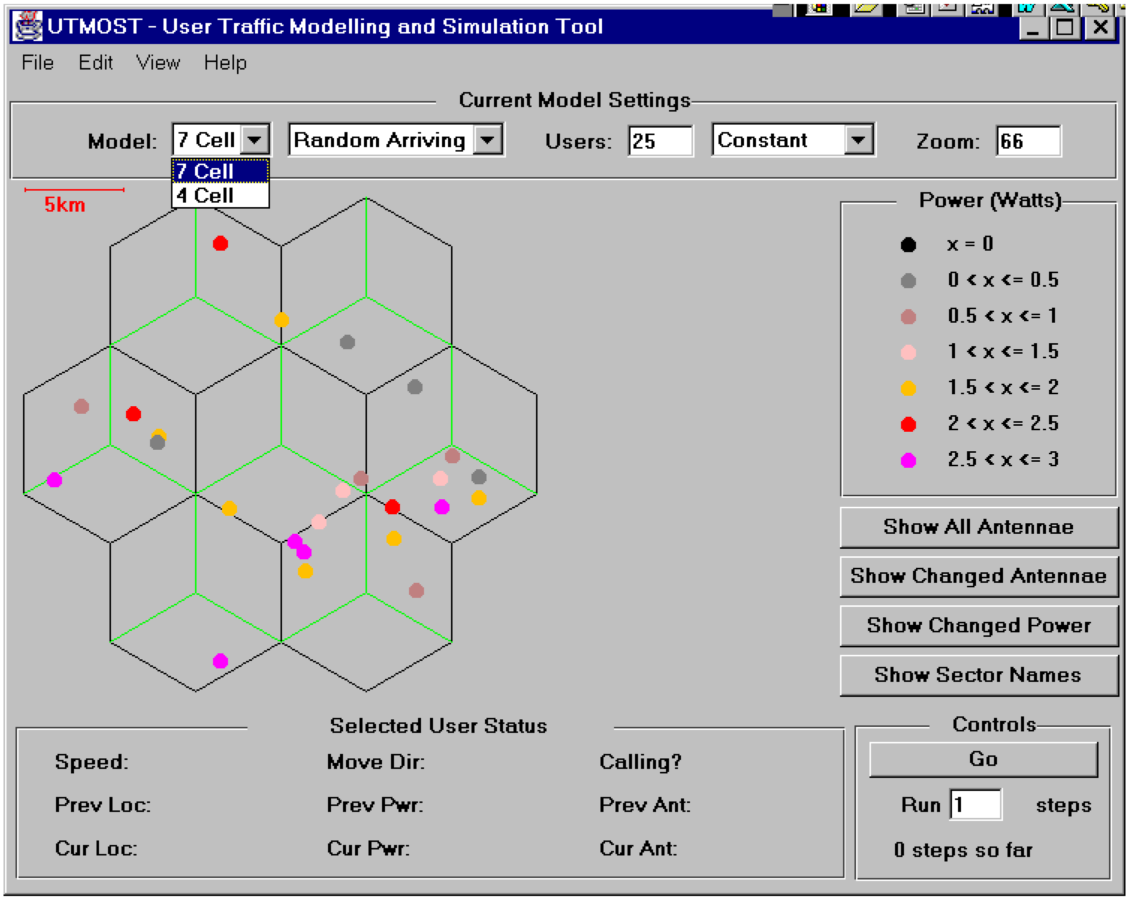


Figure 2. Model Pattern Selection Pop-up Menu

the model, or may start from the perimeter or center of the model. Users may be constant in the model or they may regenerate after reaching their destination.

Constant user behavior means that once a user is initialized, that user never changes its identity. That is, if the user travels outside the boundaries of the model, the user just continues on its path, or if the user reaches its destination in the center of the model, the user just stays at the center.

Regenerating user behavior means that once a user reaches its destination this user is re-initialized. That is, the user is given totally new coordinates, power levels, and direction of travel.

## 2.2 UTMOST GUI Implementation

UTMOST is a Java application, so it requires a Java Virtual Machine (JVM) in order to be executed. It has been successfully run on Microsoft's Win95 and WinNT, Sun's Solaris, Digital's Ultrix and SGI's IRIX operating systems. It requires JDK 1.0.2 or greater version of the JVM.

## 2.3 The User Interface Design

To execute UTMOST, one must be in the directory in which the UTMOST class files are (~racewin/src) and one may need to set the Java CLASSPATH environment variable to include the current directory ("setenv CLASSPATH ." for Unix csh.) To run UTMOST type "java UTMOST". The window shown in Figure 1 should then appear.

Next you may wish to select other initial model settings than the defaults shown. Select the model pattern by pressing the first "Model:" pop up menu and select "7 Cell" or "4 Cell" pattern. See Figure 2.

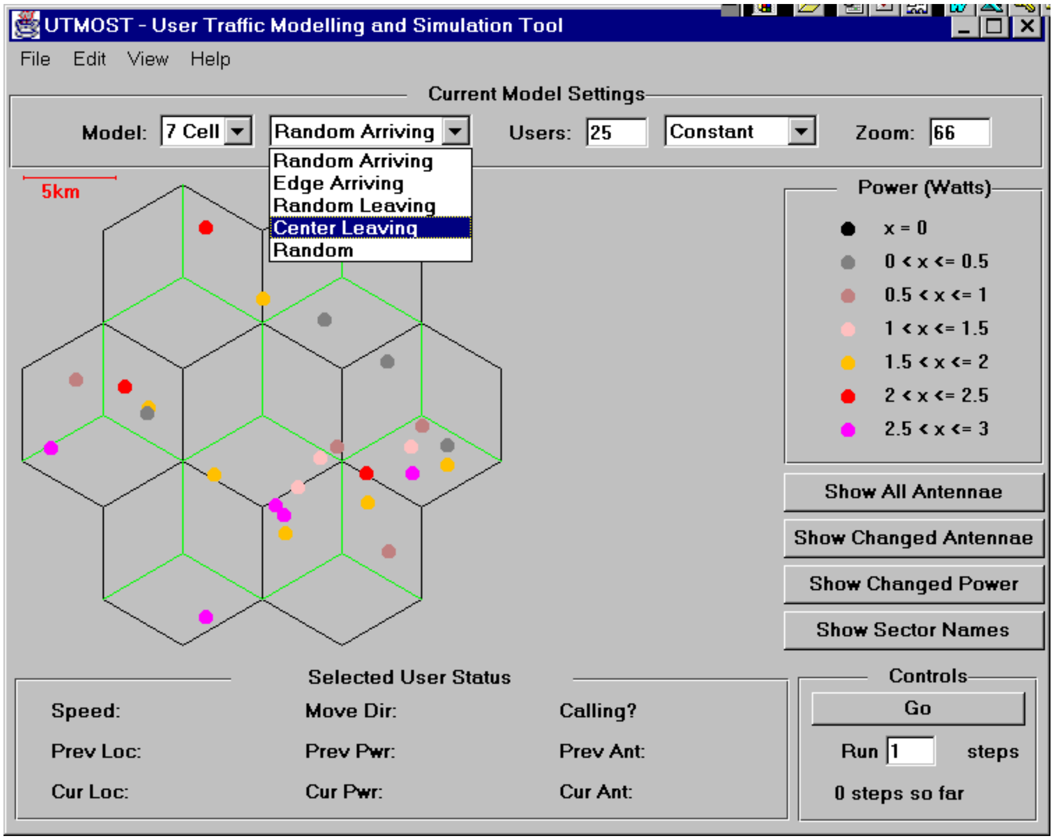


Figure 3 Model Type Selection Pop-up Menu

The user can then select the type of model to run as shown in Figure 3. Table 1 list the model types currently supported in UTMOST.

**Table 1 UTMOST Model Types**

| Pattern | Model Type | Description |
|---|---|---|
| 7 Cell | Random Arriving | Users start from random locations inside the 7 Cell pattern and all move towards the center. |
| 7 Cell | Edge Arriving | Users start from the edge (perimeter) of the 7 Cell pattern and all move towards the center. |
| 7 Cell | Random Leaving | Users start from random locations inside the 7 Cell pattern and all move towards the outside edge of the model in a straight line from the center. |
| 7 Cell | Center Leaving | Users start from the center of the model and all move towards the outside edge of the model in a straight line from the center. |
| 7 Cell | Random | Users start from random locations inside the 7 Cell pattern and move in random straight line directions. |
| 4 Cell | Random | Users start from random locations on the horizontal center line of the |

| Pattern | Model Type | Description |
|---|---|---|
| | Arriving | 4 Cell pattern and all move towards the right. |
| 4 Cell | Arriving to Right | Users start from the far left edge of the 4 Cell pattern on the center line of the 4 Cell pattern and all move towards the right. |
| 4 Cell | Random Leaving | Users start from random locations on the horizontal center line of the 4 Cell pattern and all move towards the left. |
| 4 Cell | Arriving to Left | Users start from the far right edge of the 4 Cell pattern on the center line of the 4 Cell pattern and all move towards the left. |
| 4 Cell | Random | Users start from random locations on the horizontal center line of the 4 Cell pattern and move in random left or right directions. |

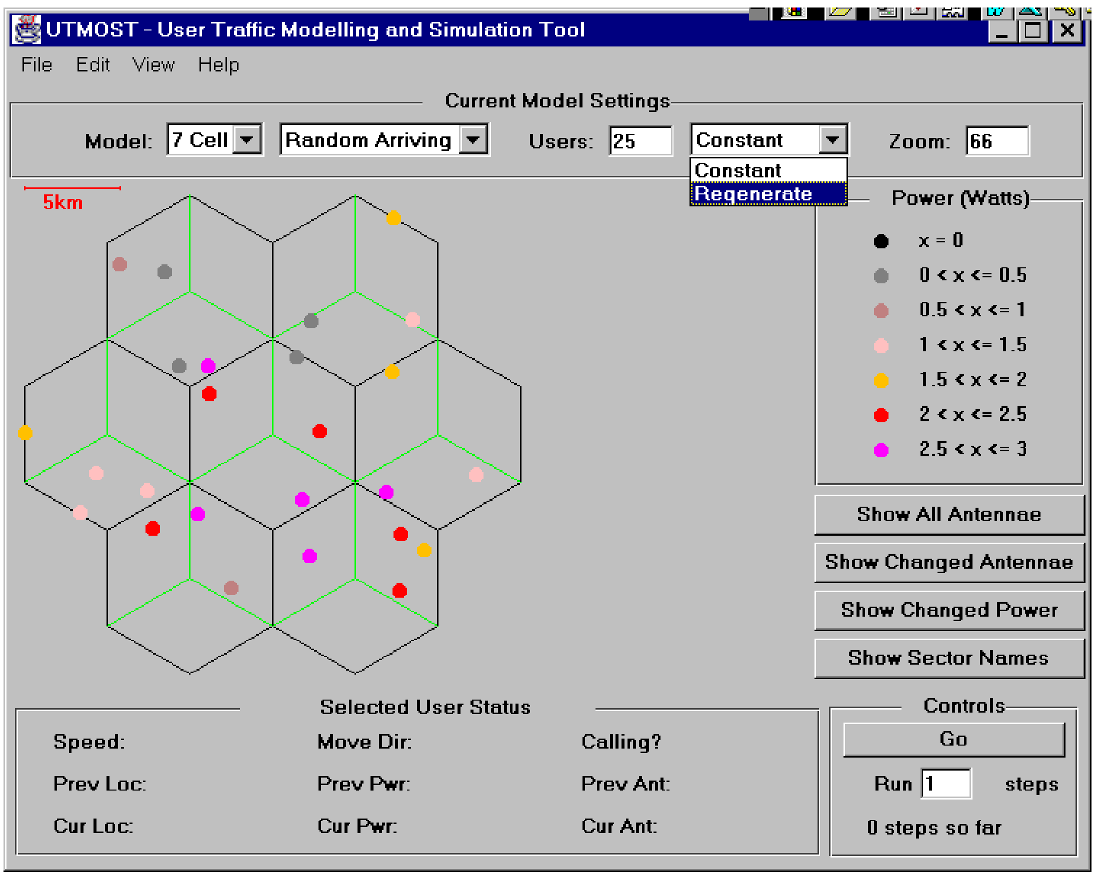


Figure 4 User Behavior Selection Pop-up Menu

The selection of the number of users to be modeled is done by typing in the user field. You can also select the two different kind of user behavior. Constant means that once a users begins its movement, if it moves outside the pattern or reaches its destination, it doesn't move. Regenerate means that the user will be regenerated with a completely new set of attributes when it moves outside the pattern boundaries or reaches its destination. Figure 4 shows the pop-up menu for the user behavior selection.

Once the model has been defined, press "Go" to run 1 or more steps. To run more than 1 step, change the "Run steps" number. The current power level or cell antenna assignments can be displayed by pressing the corresponding buttons listed in Table 2. They are toggle buttons. The display of the assignments can be removed by pressing the same button again.

**Table 2 Show/Hide Buttons in UTMOST**

| **Button Name** | **Description** |
|---|---|
| Show/Hide All Antennae | The antenna assigned to each user is indicated by a black line from the user to the antenna. |
| Show/Hide Changed Antennae | After any step in which a user changes its assigned antenna, a green line will be shown from the user back to its previous assigned antenna and a red line will be shown from the user to its newly assigned antenna. |
| Show/Hide Changed Power | Any user whose power has changed will be shown with a red circle around the user dot. |
| Show/Hide Sector Names | The sector names will be shown in each sector. A name of 5B means the cell named 5 and sector named B. |

The display of changed antenna assignments help the designer understand and verify the assignment results generated by the dynamic channel assignment procedures and the simulation results generated by network resource discrete event simulators. It actually helps to verify the computation results of UTMOST on the current sector of a mobile station, and times and places of the boundary crossing.

Figure 5 shows the graphic representation of all the mobile station-sector antenna assignments. The lines represent the antenna assignments typically forms star shape clusters. An inefficient assignment can be spotted very easily as long lines across the cell layout area.

The "Show Changed Antennae" function was designed to verify the results of hand-off algorithms. During the simulation, when a hand-off occurred, it was represented as two lines, the red

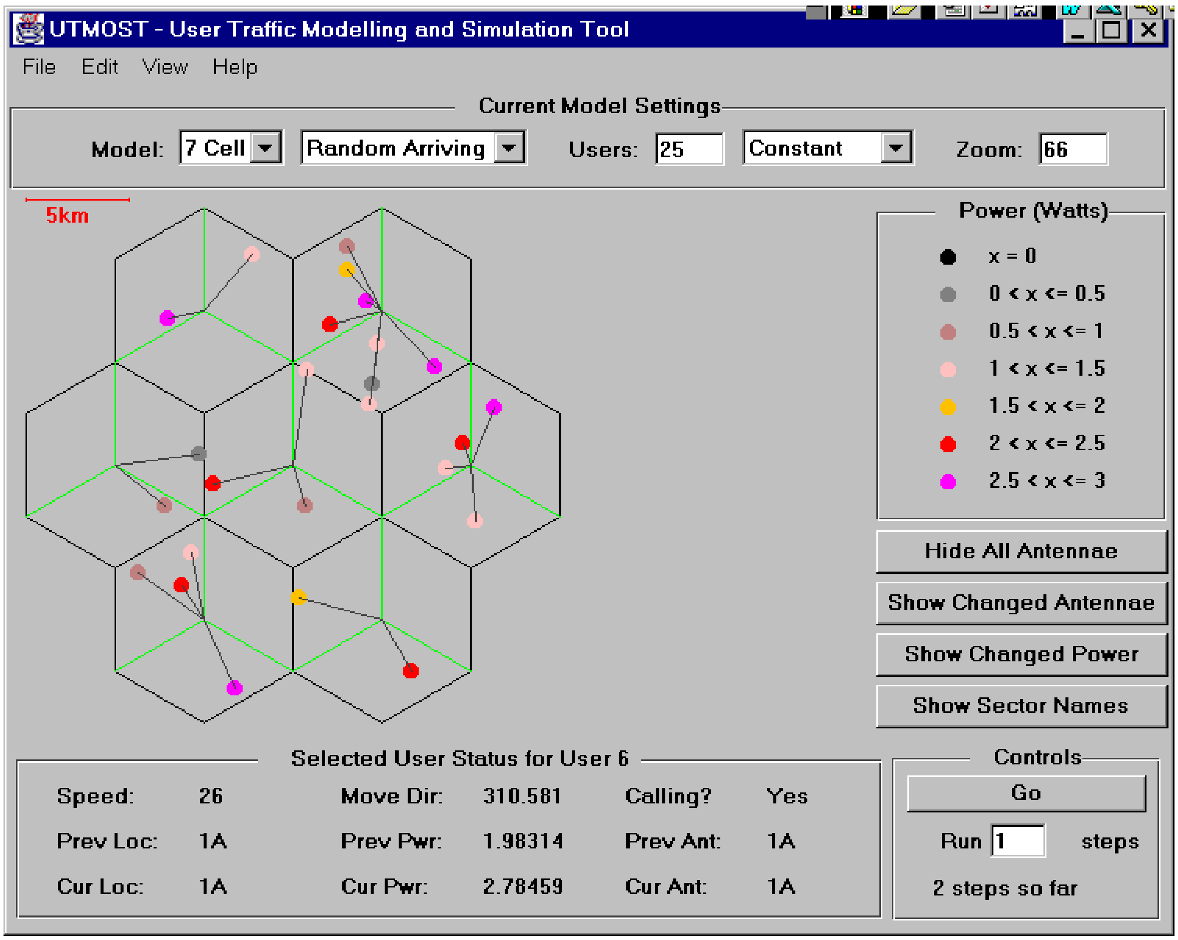


Figure 5. Antenna assignments represent as star clusters.

line shows the new antenna assignment of the mobile station while the green line shows the old antenna assignment. Figure 6 show the UTMOST display when the "Show Changed Button" is pressed."

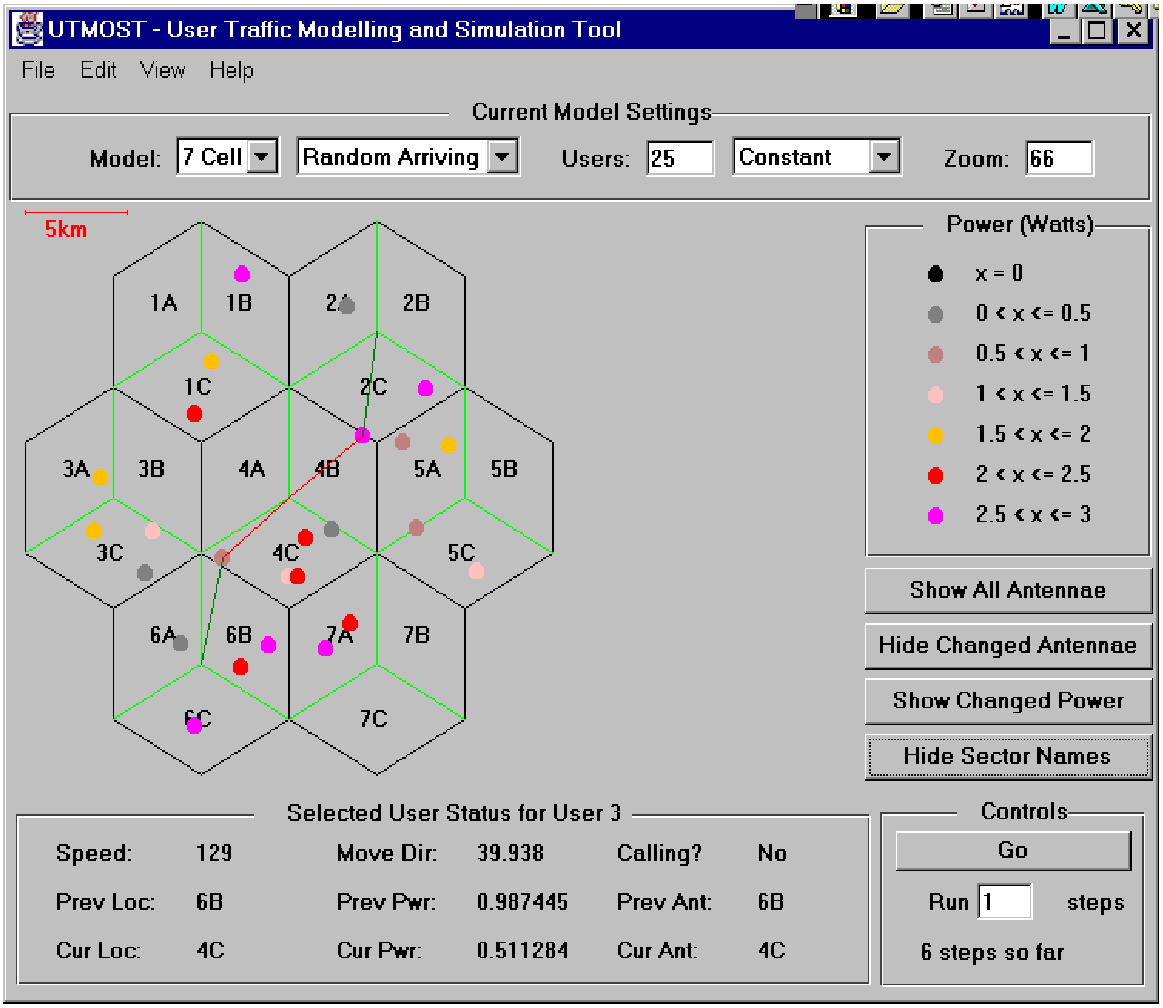


Figure 6. Show changed antenna for verifying hand-off results.

Figure 7 shows the display of UTMOST when the "Show Changed Power" button is pressed. It only shows those mobile stations that have changed power levels during the last simulation step.

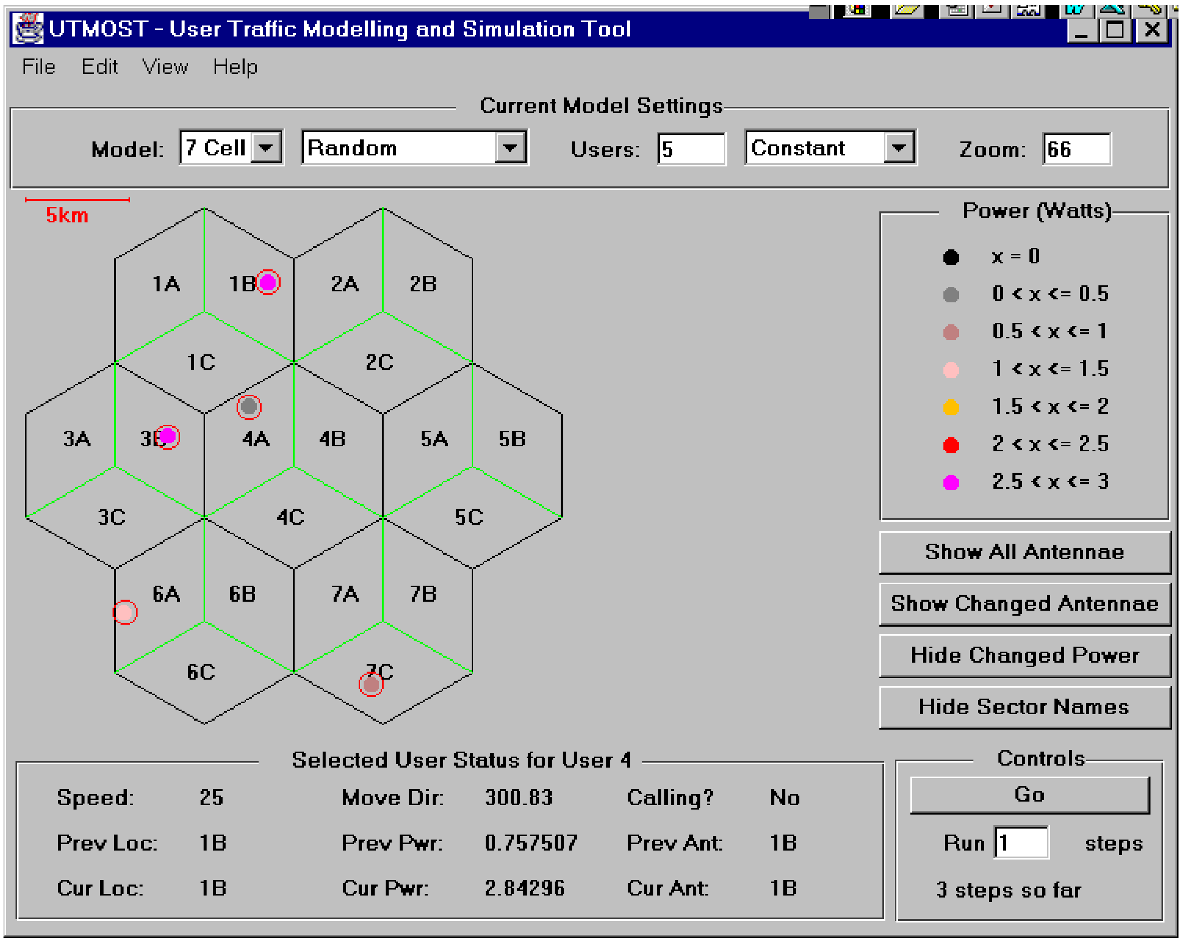

Figure 7. Show the mobile stations which just change their power levels.

Every time "Go" is pressed in UTMOST, all users' data are updated. The current state of each user is stored in an activity data file. If a user crosses a boundary from one sector to another, then an entry for this user is made in the boundary data file. This information includes exactly where the user crossed the boundary and at what time it crossed.

## 3. Computation Issue in the design of UTMOST

### 3.1 Boundary crossing

In order to calculate exactly when and where a user crosses a sector boundary, some boundary crossing algorithms needed to be created. The algorithm involves tracking all sectors in a LineSegment array. The LineSegment class includes the two (x, y) coordinates which define the line segment in addition to the slope (m) and intercept (b) for the line. That is, m and b are the constants in the equation mx + b = y which defines the line on which are the two coordinates. The application specific addition to this LineSegment class is a two element array of integers. This array is the sectors array which will keep track of the 1 or 2 sectors which share this line segment.

A user is known to have crossed a boundary when its current sector location is different from its previous sector location. With the LineSegment array to keep track of all lines in the pattern, the exact boundary which the user crossed may be calculated using a LineSegment method called intersects().

The brute force method looks at each line in the LineSegment array and checks for segment intersection. The more optimum method uses the knowledge that a user is traveling at certain direction and must cross an adjacent sector's line segment to get to its new location. First calculate the four directions from the mobile stations to the four corners of the current sectors. Then check which two corner directions the traveling direction falls between. The selected two sector corner decides the line segment which the user will cross the sector boundary. The intersection of the selected line segment and the traveling direction line is the boundary crossing point. The distance between the current location and boundary crossing point divided by the traveling speed yields the boundary crossing time relative to that of the last snapshot.

### 3.2 Perimeter placement of users

One of the models is the "7 Cell, Edge Arriving" model. This model has all users starting from the perimeter edges of the 7 Cell pattern. To place users in this initial state, UTMOST uses the LineSegment Array to find all sector lines that only have one sector in the sector array. That is, UTMOST finds all lines for sectors which do not share with another sector. Once these edges are found, the users are randomly assigned to one of these edges. Then a random (x, y) location is found on that edge and this location is used for the user's location.

### 3.3 Custom inside function

An important piece of user information needed is the current sector location of the user. Every time a user moves, that user may move into a new sector (boundary crossing) or may stay in the same sector. The first approach to calculate the sector location was to use a Java built-in polygon method called inside().This proved to be insufficient. UTMOST also handles the special case where a user is exactly on an edge of a sector.

During the development of UTMOST, it was quickly learned that the polygon functions based in Java are for on-screen graphics and therefore, only use integer (x, y) coordinates. This was not accurate enough for calculating user locations in UTMOST. Instead, a custom calcInside() method was developed to handle floating point (x, y) and floating point sector coordinates.

The Java Polygon class consists of two arrays of coordinates and an integer variable indicating how many points there are. See the Polygon constructor below:

```
public Polygon(int xpoints[],int ypoints[],int npoints)

Constructs and initializes a Polygon from the specified
parameters.
Parameters:
xpoints - the array of x coordinates
ypoints - the array of y coordinates
npoints - the total number of points in the Polygon
```

The two arrays are for the x and y coordinates respectively. That is, the 0 index of the x array is paired with the 0 index of the y array. The arrays are arrays of integers! This is the main problem with the Polygon class.

Users in UTMOST are represented by an (x, y) coordinate. The coordinate is made up of floating point numbers. Sectors in UTMOST are defined by an (x, y) point representing the base station for that sector, a direction at which the antenna for this sector points, a radius of the cell of which this sector is a part, and the correct offsets from the base station to create the sector. All of the points, radius and offsets are floating point values, also.

In trying to use the built-in Java Polygon class and casting the coordinates, etc., to integer values introduced error which caused some users boundary cross to be discovered at the wrong execution (Go) step of the model. Boundary crosses from one sector to another must be found in the modeling step in which they happen. This is why the calcInside() method needed to be developed.

The algorithm used in calcInside() is based on the fact that the sum of all angles inside a triangle should equal 180 degrees.

First a sector is divided into two triangles. The first triangle uses the first three points in the polygon defining the sector. That is, the 0th, 1st and 2nd points of the polygon definition array are used. The second triangle is the 2nd, 3rd and 4th points of the polygon. Then, using the same algorithm on each triangle, all of the inside angles are calculated as shown in Figure 8. If point (x, y) is inside the triangle, the sum of A through F angles will be 180 degrees.

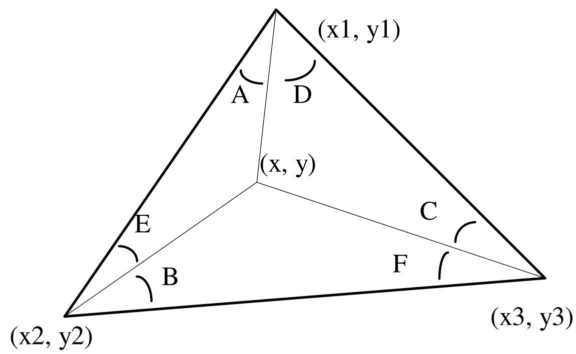


**Figure 8. Triangle Angles for calcInside() Method**

If (x, y) is outside the triangle, the sum of A through F will be greater than 180 degrees. A point outside the triangle would look like Figure 9.

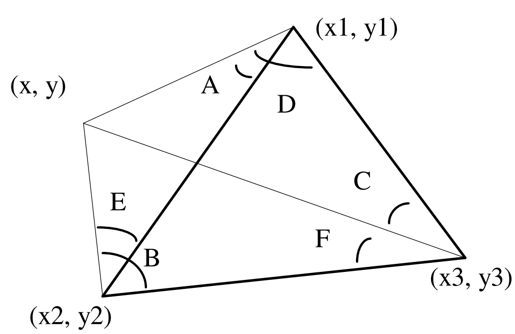


**Figure 9. Triangle with (x, y) Outside the Triangle**

### 3.4 Visible sectors calculations

One of the UTMOST output is a visible sectors string for each mobile station. This string is calculated just before user activity data is written out to the activity file. That is, after the user's next position is calculated, the new user information is used to determine the sectors which are now visible to this user. The law of cosines is used to calculate the angle between the user's location and each of the three lines leaving each base station. The angle made with the visible sectors lines will be less than 120 degrees.

This is done by calculating visible sectors to a user by using the base station coordinates and the sector lines which share the base station locations. That is, given a base station (bx, by) coordinate, there are three line segments which project out from this point. Then, using the line from the user coordinate (x, y) and the base station (bx, by) and each of the three base station line segments, the angle is calculated. The sector which is visible to the user is the sector who owns the two lines which have an angle with the point (x, y) which is less than 120 degrees. See Figure 10.

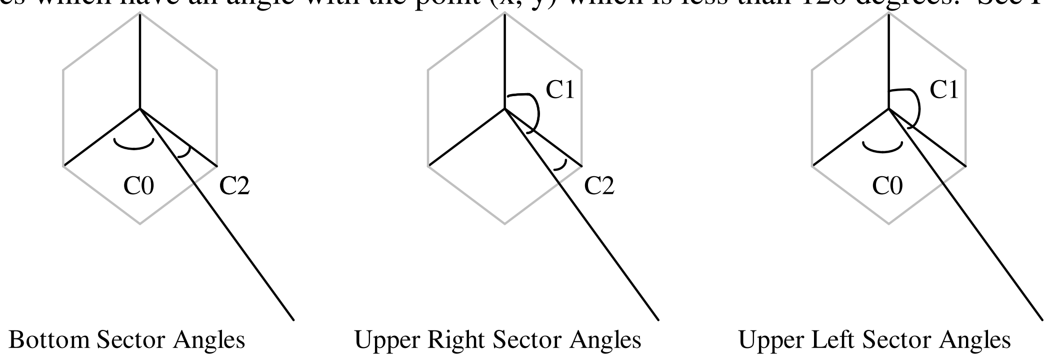


**Figure 10. Visible Sector Angles**

## 4. Summary

We have presented a wireless network traffic modeling and simulation tool with graphic user interface for visualizing the cell layout and user traffic patterns, and for facilitating the verification of the power and cell site assignment results. It also generates data for dynamic channel assignment packages and wireless network discrete event simulators. We are currently extending the tool's capability to model large number of cells and hierarchical cellular overlays.

## 5. References

[1] FCC, "Amendment of the Commission's Rules to Establish New Personal Communications Services," GEN Docket No. 90-314, Oct. 22, 1993.

[2] I. Katzela and M. Naghshineh, "Channel Assignment Schemes for Cellular Mobile Telecommunication System," IEEE Personal Communications, Vol. 3, No. 3, June 1996, pp. 10-31.

[3] Mischa Schwartz, "Network Management and Control Issues in Multimedia Wireless Networks," IEEE Personal Communications, Vol. 2, No. 3, June 1995, pp. 8-16.

[4] Lon-Rong Hu and S. S. Rapport, "Personal Communications Systems Using Multiple Hierarchical Cellular Overlays," IEEE JSAC, Vol. 13, No. 2, Feb. 1995, pp. 406-415.

[5] Chih-Lin I, L. J. Greenstein, and R. D. Gitlin, "A Microcell/Macrocell Cellular Architecture for Low- and High-Mobility Wireless Users," IEEE JSAC, Vol. 11, No. 6, Aug. 1993, pp. 885-891.

[6] Miltiades E. Anagnostou, G. C. Manos, "Handover Related Performance of Mobile Communication Networks," Proceedings of IEEE 44th Veh. Tech. Conf., VTC 94, pp. 111-114.

[7] Djamal Zeghlache, "Aggressive Handover Algorithm for Mobile Networks," Proceedings of IEEE 44th Veh. Tech. Conf., VTC 94, pp. 87-90.

[8] Rajiv Vijayan and J. M. Holtzman, "A Model for Analyzing Handoff Algorithms," IEEE Trans. On Vehicular Technology, August, 1993.
[9] Leung, Kin K., "Traffic Models for Wireless Communications Networks," IEEE JSAC, Vol 12, No. 8, Oct. 1994.
[10]Foley, James D., Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics: Principles and Practice, Addison-Wesley Publishing Company, 1990.
[11] Geary, David M. and Alan L. McClellan, Graphic Java-Mastering the AWT, The SunSoft Press, 1996.
[12] Jackson, Jerry R. and Alan L. McClellan, Java by Example, The SunSoft Press, 1996.

| BIBLOGRAPHIC DATA SHEET | 1.Report No. EAS-CS-97-1 | 2. | 3. Recipient's Accession Number |
|---|---|---|---|
| 4. Title and Subtitle **UTMOST: A Traffic Modeling Tool for Wireless Network Design** | | | 5. Report Date **April, 1997** |
| | | | 6. |
| 7. Author(s) **C. Edward Chow and Heidi McClure** | | | 8. Performing Organization Report No. **EAS-CS-97-1** |
| 9. Performing Organization Name and Address **Department of Computer Science University of Colorado at Colorado Springs Colorado Springs, CO , 80933-7150** | | | 10. Project/Task/Work Unit No. |
| | | | 11. Contract/Grant No. **CASI FY97** |
| 12. Sponsoring Organization Name and Address **Colorado Advanced Software Institute 2270 South 8th Street M.S. 4232 Louisville, Colorado, 80028-4232** | | | 13. Type of Report & Period Covered |
| | | | 14. |

15. Supplementary Notes

16. Abstracts

**In this paper we present the design and implementation of a user traffic modeling and simulation tool for wireless network design. It has a graphical user interface for displaying the cell layout, user traversing patterns, the current power and cell site assignments. For each simulation step, it generates an output file of the current location of the users, their power levels and cell assignments, and the boundary crossing times and locations. It also computes the visible sectors for each mobile user. It can interact with dynamic channel assignment optimization packages and facilitate the verification of the assignment results. It outputs event traces to drive discrete event simulators for the cell size determination and the comparison of efficient hand-off algorithms.**

17. Keywords and Document Analysis. 17.a Descriptors

**Wireless Network**
**Network Design**
**Graphic User Interface**
**Java Applications**
**Network Traffic**
**Modeling and Simulation**

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

| 18. Availability Statement | 19. Security Class (This report) | 21. No. of Pages 13 |
|---|---|---|
| | 20. Security Class (This page) | 22. Price |