

Path Estimation Using Cellular Handover

Mathé Young Mosny

April 17, 2006

Advisor: Professor Alain L. Kornhauser

Submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Engineering
Department of Operations Research and Financial Engineering
Princeton University

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Mathé Young Mosny

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Mathé Young Mosny

Acknowledgements

Thank you first and foremost to my family. Mom, you have shown me the meaning of true kindness with your constant love and unending compassion. Ba, you have taught me what it means to be a man. Alaina, I am more proud of you than you could ever imagine, and I cannot wait to enjoy with you the fantastic success that you will soon experience. Aimi, you are the reason for any and all of my achievements and I will forever be in your debt; I love you with all of my heart.

Thank you to my advisor Professor Kornhauser. Your guidance and friendship throughout the years will truly give me cause to reflect on my Princeton career with fondness.

Thank you to my friends. Rob, you are my best friend; there is nothing I can say here that you do not already know. Ariel, thank you for getting me through the all-nighters and keeping me sane. Train, just try to remember my name when you're a superstar. John, you've been down for anything since the beginning, and I respect you deeply for it. J, it makes me happy to know that you are living it up on the west coast. Athan, I have never met a man who was either loved or hated by more people in my life. Andy, you've always been my boy. Luv, the best times are yet to come.

Finally, I would like to thank Emily, my unending source of happiness. You have given me the best days of my life. I live only to repay you.

Thank you all,

Mathé Young Mosny '06

Table of Contents

| | |
|---|----|
| Chapter 1: Introduction..... | 5 |
| Chapter 2: Current Protocols..... | 9 |
| Chapter 3: Time and Location Value..... | 21 |
| Chapter 4: Building a Model..... | 44 |
| Chapter 5: Alternatives and the Financial Case..... | 62 |
| Bibliography and References..... | 66 |
| Appendix A..... | 69 |

*“Mobile phones are the only subject on which men
boast about who’s got the smallest.”*

--Neil Kinnock

1 Introduction

Twenty years ago, cellular telephones as they are known in the conventional sense were entirely non-existent. Although a very small minority of technophiles and military personnel had access to portable or wireless telephony, the majority of the world had never heard of cellular phones, much less witnessed their operation firsthand. With the absence of the Internet and E-Mail, individuals were relegated to communicate using only “plain old telephone service”, or POTS as it is known today. Land-line POTS carriers would employ a host of technologies to route callers to one another, the main one being the ever-important switch. Groups of switches bundled together in each municipality were called switchboards and used electronic routing systems to instantly connect call dialers to call receivers. Communicating between large distances was an

easy task, even though users of land-line telephones were tethered physically to their respective ends. People had not yet experienced the freedom of wireless communication.

In the early to mid-1980s, ultra high frequency (UHF) satellite radio bands from 400 MHz to 1.5 GHz were launched to facilitate mobile communication among pedestrians (Gibson 1071). Though many cellular systems existed at the time, the dominant standard in the United States was the North American Advanced Mobile Phone Service, or AMPS architecture. The AMPS system would, after several other iterations, be replaced by first, second, and eventual third-generation TDMA (time division multiple access) and CDMA (code division multiple access) protocols. The Conference of European Postal and Telecommunications Administrators (CEPT) established another standard, the GSM standard, for European communications in 1982 (Gibson 1072).

The advent of new wireless standards established in the 1980s brought about explosive growth in consumer support and private investment in cellular technology over the next twenty years. Early providers of both satellite and what Gibson describes as “terrestrial mobile radio systems” would see usage increase by up to 50% annually over the next years, and by 1994 there were about 8 million cellular telephone subscribers in the United States alone (1072). While numerous terrestrial (using Earth-based cellular towers) and satellite wireless protocols were developed over the rapid expansion phase of the 1980s and 1990s, they all shared several methods of operation that are still in use today. As terrestrial mobile systems will be the focus of this paper, Gibson does a very good job in describing the evolution of the land-based cellular network:

In a terrestrial mobile radio network, a repeater was usually located at the nearest summit, offering maximum service area coverage. As the number of users increased, the available frequency spectrum became unable to handle the increased traffic, and a need for frequency reuse arose. The service area was split into many small subareas called cells, and the term cellular radio was born ... The tradeoffs between real estate availability (base stations) and cost, the price of equipment (base and mobile), network complexity, and implementation dynamics dictate the shape and the size of a cellular network. (1073)

As terrestrial cellular telephone service grew in popularity, so too did the need to construct base broadcasting towers and cellular repeaters to provide coverage in increasingly remote areas of the country. And while the sound and coverage quality of satellite systems far exceeded those of their terrestrial counterparts, the ubiquity of land-based networks would ensure the future of cellular communication.

Satellite mobile networks, however, did allow for one feature that has not been possible up to this point in terrestrial mobile networks: A robust and accurate Global Positioning System (GPS) whereby the physical location – and by deduction, speed – of client users could be obtained and recorded. In the mid 1990s, the United States Department of Defense launched a system of 18 to 24 GPS satellites to produce specific radio signals from which “an intelligent microprocessor-based receiver [would] be able to ... accurately determine its own three-dimensional position, velocity, and acceleration worldwide” (Gibson 1074). Unfortunately, terrestrial mobile networks did not until very recently possess this functionality; users of traditional cellular service were relegated to simple analog/digital voice communications with no positioning features. Some alternatives exist within the terrestrial sphere that attempt to provide the user with some location-based functionality, but they are for the most part too expensive or cumbersome for widespread use and private implementation.

A New Use for Cellular

Looking ahead ten years, we find ourselves in the heart of the information age where increased information of any sort often translates to enlarged profits and bigger bottom lines. Google has transformed itself into a \$120 billion company mainly by providing advertising relevant to the information found on a given site (Yahoo! Finance: GOOG). Smaller, niche companies such as DailyCandy are being valued at over \$100 million simply to provide information on clothing sales to women in major metropolitan

areas across the country. Surely, quality advertiser/consumer and consumer/advertiser information delivery comprise a booming industry.

Today, numerous consumer GPS hardware and software clients exist to provide users with location and speed data and are often packaged with trip-planning software or route optimizers. Regrettably, all personal GPS units require satellite GPS protocols to operate successfully. That is, no current consumer or commercial GPS solution uses any form of terrestrial radio or cellular system to locate clients. This shortcoming of current techniques is unfortunate primarily for two reasons: First, satellite reliance typically increases operation cost by several orders of magnitude. This cost increase occurs because the user is forced to purchase a separate GPS satellite receiver if one is not built into the client machine or computer. Second, GPS satellite dependence excludes any use of mobile cellular telephones for user location. Considering that over one billion cellular phones are in use worldwide (NetworkWorld), this is strikingly large demographic to overlook.

This paper will explore different methods to produce client location and speed information using only terrestrial mobile radio protocols. By removing the need to explicitly use GPS satellites to obtain the same information, one can effectively reduce the associated cost by a factor of hundreds or even thousands. Furthermore, by combining acquired client location and speed data with path-recognition algorithms and relevant time-stamping, one can produce real per-user path and trip choice data. Finally, using cellular tower client handover data with proper tower-location algorithms can produce surprisingly accurate tower location approximations, eliminating the need for third party field research. While focusing primarily on methods involving tower handover, the paper will discuss current handover-less alternatives, and examine the financial case for providing a GPS-like service relying only on handover.

"I don't own a cell phone or pager. I just hang around everyone I know, all the time. If someone wants to get ahold of me, they just say, 'Mitch,' and I say, 'what?' and turn my head slightly.

--Mitch Hedberg (1968-2005)

2 Current Protocols

The brisk growth of cellular mobile phone use during the 1980s was unexpected and, as a result, unorganized. Since no cellular standards were in place, local cellular services were ad-hoc at best and far from reliable. Using exclusively analog technology, phone conversations were carried out with varying quality levels while calls were often dropped unpredictably. Being a relatively new technology, the Federal Communications Commission (FCC) was relatively reluctant in embracing cellular telephone as a complement or replacement to traditional land-based telephony. In their stance, the FCC allocated only a very small portion of the electromagnetic spectrum traditionally used for wireless communication to the cellular realm. The combination of these factors severely jeopardized the future of the cellular telephone. Redl, Weber, and Oliphant do

well in summarizing the major limitations of the state of the technology at the end of the 1980s:

(1) severely confined spectrum allocations; (2) a perception among more sophisticated users ... that the systems were limited in usefulness because of annoying sounds and interference as the mobiles moved about ... (3) inability to substantially lower the cost of mobile terminals and infrastructure, and (4) incompatibility among the various analog systems, especially Europe, thus preventing the subscriber from using his or her phone abroad. (4)

From the beginning of the 1990s onward, the cellular landscape began to change drastically in order to accommodate increased usage numbers and higher call quality requirements. Mobile radio shifted away from analog operation to digital operation, and two dominant cellular standards emerged; the United States adopted the TD/CDMA (Time Division/Code Division Multiple Access) protocol while Europe and the rest of the developed world favored GSM (Global System for Mobile Communications). The two protocols ensured the future of cellular communication while their variants are still in use today.

TD/CDMA

Both the TDMA and the CDMA standards were developed to help squeeze as much performance as possible out of the limited spectrum that the FCC had allotted for cellular communications. And although both use different approaches towards this end, CDMA has become the dominant standard in the United States today by proving its robustness in various call situations. Also, CDMA allows for “soft handover” of phone calls from one cellular tower to another, a crucial feature which helps to prevent call drops. Soft handover and its sister technology, hard handover, are of central importance to the location methods of this paper and will be explained in further detail.

By the end of the 1980s, all cellular carriers relied on analog signal transmission to connect cellular calls and consequently exhausted all available spectrum to capacity. TDMA, or Time Division Multiple Access, was invented as a technology to digitally transmit signal and allow “a number of users to access a single radio-frequency (RF)

channel without interference by allocating unique time slots to each user within each channel” (IEC: TDMA). In other words, TDMA would allow several cellular conversations to be simultaneously transmitted over one channel and properly decoded by handsets as long as each individual conversation was offset some constant amount of time. This was achievable by “bursting”, or sending small portions of each conversation across the channel at given offsets. An example of such a system involving three concurrent conversations would resemble the following:

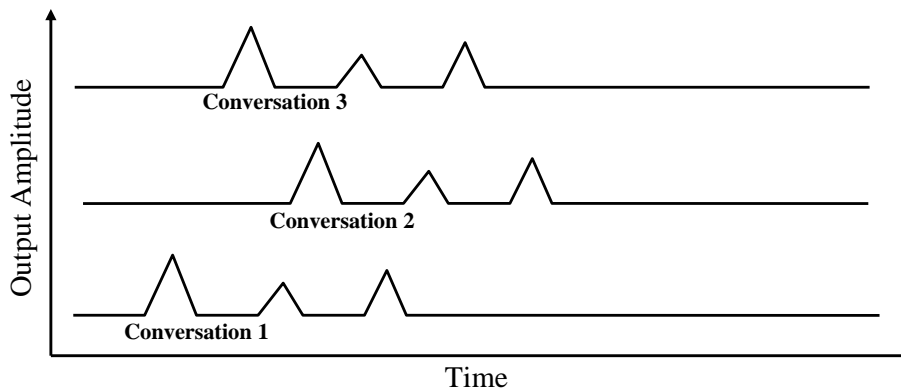


Figure 2.1. Example TDMA conversation sequence.

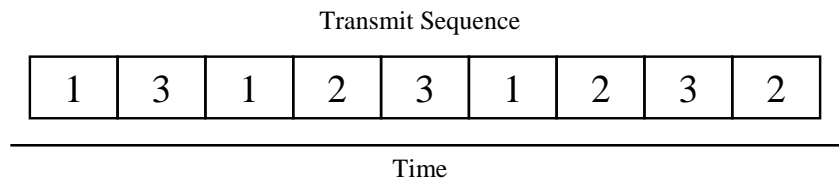


Figure 2.2. TDMA transmit sequence.

Each conversation is transmitted digitally, using bits to transmit the digital data. That is, each burst is transmitted as a series of 1’s and 0’s and can be decoded by computers at both ends. Each burst contains two separate data parts, denoted by $i \in \{1, 2\}$ sent in blocks numbered $k \in \{1, 2, \dots, K\}$. The bursts are “interleaved”, being either $l = 1$ or $l = 2$, and the overall packet number is accounted for by an index $n \in \{1, 2, \dots, N\}$. Then, Prasad, Mohr, and Konhäuser give the equation by which bits $d_n^{(k,i)}$ are transmitted as (34):

$$\operatorname{Re}\left\{\underline{d}_n^{(k,i)}\right\} = \frac{1}{\sqrt{2}}\left(2b_{-1,n}^{k,i} - 1\right), \operatorname{Im}\left\{\underline{d}_n^{(k,i)}\right\} = \frac{1}{\sqrt{2}}\left(2b_{-2,n}^{k,i} - 1\right) \quad (2.1)$$

Where,

$$b_{-l,n}^{(k,i)} \in \{0,1\} \quad (2.2)$$

Whereas TDMA multiplexes several conversations by using different respective time delays, CDMA, or Code Division Multiple Access, assigns each conversation a special random code, while the conversations are eventually decoded according to each assigned code. An example involving three simultaneous conversations using CDMA coding would resemble the following:

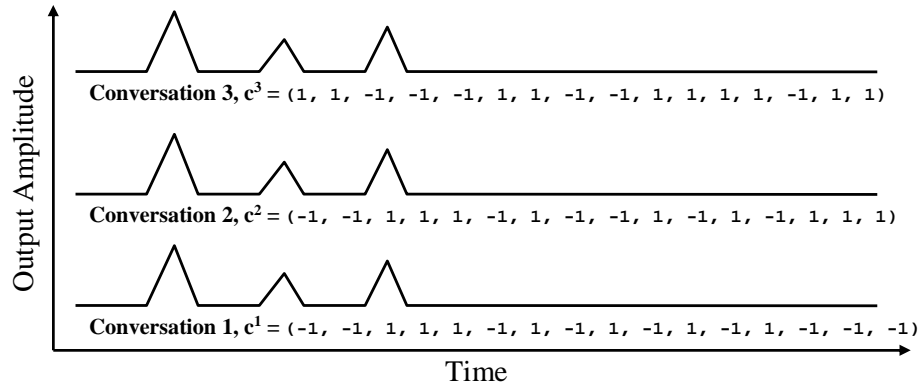


Figure 2.3. CDMA transmit sequence.

The CDMA code $\underline{c}^{(k)}$ associated with each conversation k is generated as follows, as Prasad, Mohr, and Konhäuser describe on page 35: First, the code itself must be of length $Q \in \{1, 2, 4, 8, 16\}$. Then each element $c_q^{(k)}$ of the code is taken from the complex set $\underline{V}_c = \{1, i, -1, -i\}$, where $i = \sqrt{-1}$. Finally, each individual code is calculated as

$$\underline{c}_q^{(k)} = (j)^q a_q^{(k)}, \quad a_q^{(k)} \in \{1, -1\}, \quad q = 1 \dots Q, \quad k = 1 \dots K \quad (2.3)$$

Although other, more complicated differences between the TDMA and CDMA standards exist, the only one of importance to this paper is the method by which they

handle tower handover. The techniques in each protocol used to perform handover will be compared shortly after the introduction of the GSM protocol.

GSM

In the beginning of the 1990s, European countries began to adopt the Global System for Mobile Communications (GSM) protocol to regulate cellular communication. Through persistence and tedious network construction and testing, GSM has established itself as the standard for every European country with wireless capabilities. The rapid expansion of GSM was not a coincidence. Rather, the proliferation of the standard in Europe occurred for three key reasons: First, according to Redl, Weber, and Oliphant, GSM facilitated the “unification of the European community in terms of political, social, and economic aspects” (13). Second, the push for GSM as a standard occurred directly after a series of European commercial wireless deregulations. Finally, private companies were founded to “develop and produce new infrastructure and mobile terminals for a market that [was] much larger than the small markets in single countries” (13). In addition, much of Europe did not possess adequate land-based telephone lines to begin with, making the jump to cellular an easy one.

According to Redl, Weber, and Oliphant, GSM did, in fact, live up to its promises and deliver:

(1) superior speech quality ... (2) low terminal, operational, and service costs, (3) a high level of security (confidentiality and fraud prevention), (4) international roaming (under one subscriber directory number), [and] (5) support of low-power hand-portable terminals. (23)

Having already covered the TDMA protocol earlier, the operation of GSM is fairly straightforward since it uses TDMA as its multiplexing method. Unlike the other TDMA and CDMA standards used in the United States, GSM does not establish explicit requirements for hardware operating under its standard. Instead, it only aimed to

“define the functions and interface requirements” (IEC: GSM) for companies seeking to produce GSM compatible phones.

GSM has gained popularity in the United States as well, having penetrated the market under the protocol PCS-1900. Although a large proportion of the population still uses the early-generation analog AMPS protocol, a growing number of people are switching to digital TDMA-based standards PCS-1900 (operating at the 1900Mhz frequency) or its competitor IS-136. Currently, the active CDMA protocol in use in the United States is standard IS-95, or cdmaOne (Prasad, Mohr, and Konhäuser 17). GSM is easily explained in depth, while some of its more exotic functions will be essential to calculations in future chapters. For these reasons, GSM will be the assumed protocol for all calculations and modeling throughout the rest of this paper.

The Structure of GSM

When describing the organization of a GSM network, perhaps the best place to begin is with the users themselves as they comprise the top-most level of operation. Located within each client phone is a small identification-bearing chip named the subscriber identity module, or SIM chip (Redl, Weber, and Oliphant 35). The SIM itself is physically read through integrated phone hardware and is used specifically to identify and authenticate each individual customer within the network. While a small number of personal phone numbers and short message service (SMS) text messages can be stored on each individual SIM chip, the most valuable SIM-stored pieces of information are the (1) authentication algorithm used, (2) authentication key (password), and (3) international mobile subscriber number, or IMSI (Redl, Weber, and Oliphant 37). Each distinct subscriber under the GSM network can be uniquely identified by their IMSI, while similar authentication algorithms and keys can be found from phone to phone.

The physical components of the GSM network are organized in adjacent groups called cells with base transceiver stations (BTSs) at the center of each cell. Contrary to popular belief, BTSs, or cellular towers, do not broadcast signal spherically originating at the tower itself and diminishing along circular contours outwards. Instead, each tower houses three separate antennas broadcasting in a 120-degree range (Ibid 31). Since a tower effectively designates the center of its respective cell, the network can be visualized as follows:

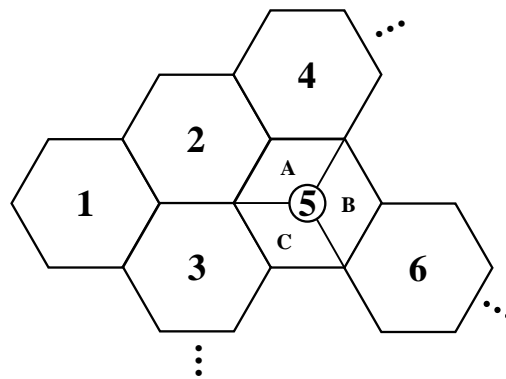


Figure 2.4. Layout of cells within a GSM network.

The individual cells above are arbitrarily labeled and can repeat across large geographical spans to accommodate growing networks and increased cell-phone usage. Cell number 5 has been drawn to reflect the three-section nature of cellular antennas with sections A, B, and C covering their respective 120-degree zones.

At this point, it should be noted that until only recently, travel from one cell to another while on an active call was not possible under the (now outdated) old GSM protocol. In other words, cellular handover had not been developed to the point to which a subscriber on an active call could drive across cell coverage zones without the call being dropped. Eventually, a method of wireless handover would be invented to solve this problem. The concept of handover is of critical importance to this paper and will be used extensively in most calculations in later chapters. Redl, Weber, and Oliphant describe the pre-handover era in depth:

The cell or network size was determined by the transmitter's power. It was not possible to have a link between two different cells, or coverage areas, since an orderly means of directing traffic (voice audio) between transmitter sites and moving mobiles was missing. It was important to select the frequency of the transmitter and receiver in the cell carefully so that there was no interference from other systems, perhaps in the next town, which would interfere with the system's local operation. (27)

Functioning cellular handover was a breakthrough for GSM at its inception.

“Handoff”, as was the term used to describe the faulty, analog method of handover, was simply slang for “dropped call” (Ibid 43). On a cursory level, its operation can be explained in a fairly straightforward manner. Controlling a small group of BTSs each are base station controllers, or BSCs. The BSCs monitor the status and frequency output of each BTS and report to the gateway mobile services switching center (GMSC), the centerpiece of the handover routine (Ibid 36).

The GMSC holds two data sets that are crucial to facilitate successful handover, the home location register (HLR) and the visitor location register (VLR). The HLR stores all relevant information relating to subscribers “belonging to the area of the related GMSC” (Ibid 37). For example, a user living in Cleveland would have their IMSI, phone number, authentication key, and service type (voicemail enabled, etc.) stored on the HLR responsible for serving Cleveland. The second register, the VLR, holds in it several pieces of information that are required for any customers that are not currently in their home zone. This information includes each user's temporary mobile subscriber identity, or TMSI, the current BTS that is actively servicing each customer, and network instructions to properly forward phone calls to each roaming user (Ibid 38). Assume, for example, that the previous Cleveland-based user took a business trip to Los Angeles. While in Los Angeles – outside of his home coverage zone – the businessman is receiving cell phone signal from a local BTS. Then, the VLR located in the GMSC that maintains that local BTS would store the businessman's TMSI while his IMSI is being stored back in Cleveland. When the businessman receives a phone call on his Cleveland phone

number, the HLR will search for a TMSI match to its IMSI and route the call to Los Angeles accordingly. The entire setup is represented below:

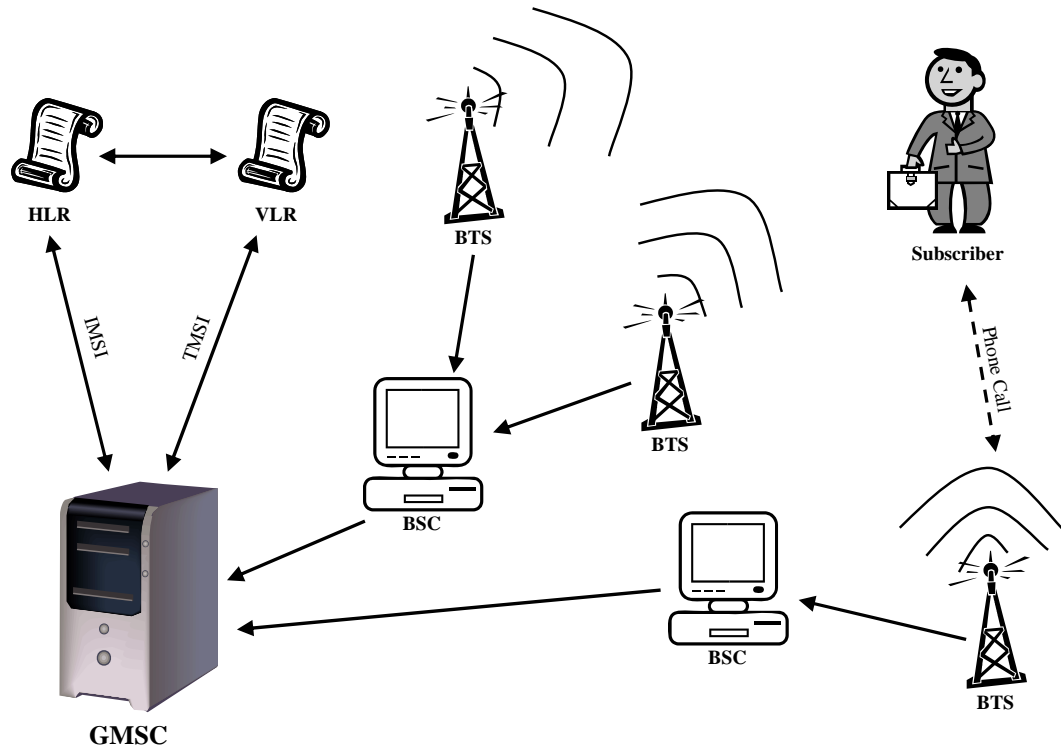


Figure 2.5. Diagram of the major parts of a GSM network and essential parts for handover. Refer to Redl, Weber, and Oliphant p. 34 for a more detailed diagram.

The fact that, for each cellular subscriber, there exists both a static set of user data in the HLR and a dynamic set of user data in the VLR is important for GSM handover to occur successfully. The handover process itself is relatively simple: When a GSM subscriber powers on her cellular phone, the handset boots up to the GSM “base channel” (Ibid 43), which is one of the network’s dedicated control channels, or DCCHs (Ibid 89). Every phone under the GSM standard boots up to the same DCCH, as it is the channel used to send information from the handset to the BTS and vice versa. Once powered up, the closest BTS –or the one with the strongest signal output to the user– provides a list of other BTSs that are nearby. The phone then performs signal measurements on the BTS that is actively serving it and on each nearby BTS on the newly-received list. The results of the signal analysis are compiled by the handset into a

“measurement report” that is sent back to the BTS at specific time intervals (Ibid 43). The measurement report propagates backwards to the BSC and finally to the GMSC when it is sent to the BTS. In effect, the GMSC always has a list of the measurement reports sent by all handsets in the area. The period of time between handset transmissions of the measurement report will be referred to as the “beacon interval” in this paper and is important in future calculations. Note that the beacon interval depends on the phone itself, so the time between measurement report transmissions depends on the manufacturer of the handset hardware. If a handset-sent measurement report indicates that another BTS would provide stronger signal, the appropriate BSC directs the BTS to transfer service to another BTS and initiate a handover. The VLR associated with the GMSC servicing the new BTS is then updated to include the handset’s TMSI, while the HLR remains unchanged. At the same time, the VLR located in GMSC associated with the old BTS is changed so that it does not include the handset’s TMSI. This transfer of VLR data across GMSCs only occurs if the user is crossing into a cell controlled by a different GMSC (i.e. crossing into another country). The process is diagrammed on the next page.

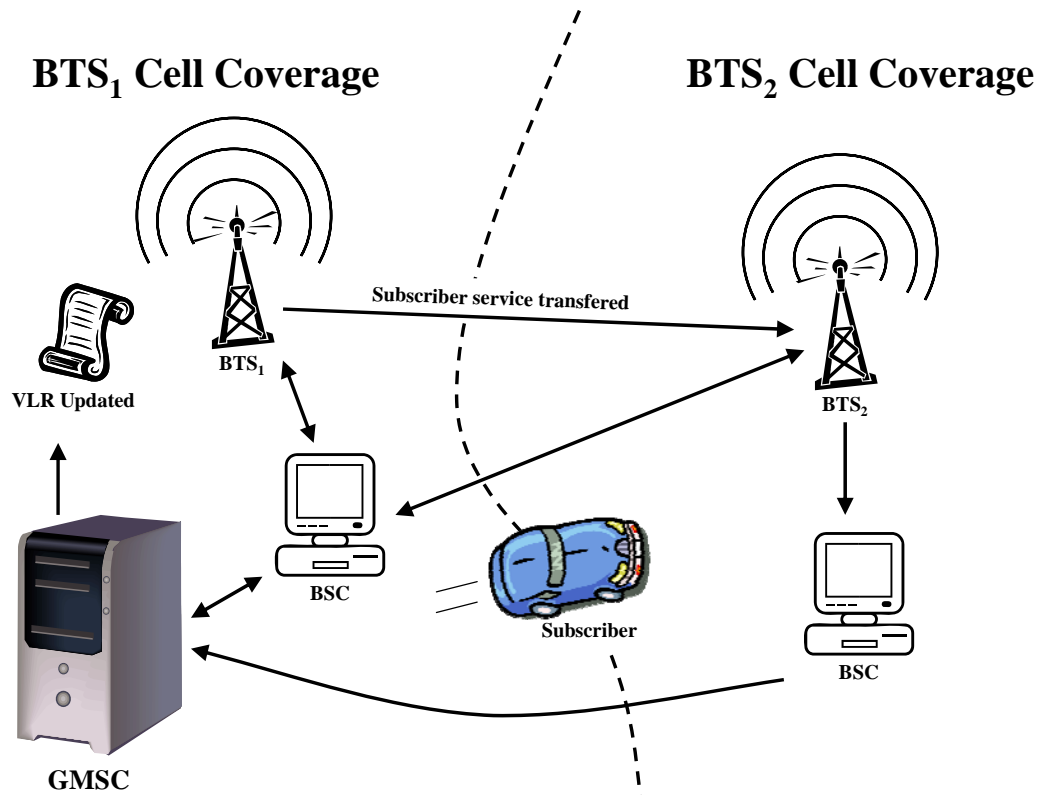


Figure 2.6. Diagram of the key components of GSM TDMA handover. Refer to Redl, Weber, and Oliphant p. 44 for a more detailed diagram.

In some cases, handset service might be transferred between different GMSCs when a user crosses from one cell into another. This might occur if a subscriber is driving from one GSM-enabled country into another, where a new GMSC is responsible for network upkeep. Refer to Redl, Weber, and Oliphant p. 45 for a detailed diagram of this behavior.

It should be noted at this point that measurement reports are sent only when the subscriber is not actively talking on a call on her handset. By contrast, when a user is active, measurement reports are not being sent over one of the DCCHs since the user is talking on another channel altogether. In these situations, handover must be handled by the appropriate BSC and occurs when the signal of another BTS becomes stronger than the signal of the current BTS (Redl, Weber, and Oliphant 43). This means that while a subscriber on an active call would almost instantaneously be transferred to the BTS with the strongest signal when handover is necessary, a passive (non-talking) user would have

to wait for their measurement report to be sent and processed to be considered for handover.

Since GSM uses the TDMA standard, handovers performed under the protocol are hard handovers unlike the soft handovers performed under CDMA specifications. Gibson defines hard handover as “characterized by temporary disconnection of the traffic channel”, and soft handover as “characterized by simultaneous communication with a subscriber by more than one base station” (1274). In other words, under GSM, a user is handed over from one BTS to another during a handover, while a user under CDMA standards is always connected to several BTSs. In general, a CDMA subscriber is connected to around 3 different BTSs at all times with capacity to connect to up to 6 different BTSs (Cooley).

“I like to drink to suit my location.”
--Tom Jones

3 Time and Location Value

In today’s fast-paced and frenzied marketplace, businesses aiming to provide the consumer with an added sense of convenience are being founded at a fantastic rate. Take, for example, the company TomTom NV, an Amsterdam-based company that provides consumer GPS receivers to be used in car navigation. The company raised around \$300 million in its initial public offering in May 2005, and it is now valued at over \$500 million one year later. Clearly, the public is willing to pay for something that provides real-time location information.

Enter the small engineering arm of a large consulting company named Delcan. The proprietary company National Engineering Technology Corporation (NET) is purporting to offer a new technology that can determine the location of anyone driving with a cellular telephone. Named “Cell Probe Technology”, the only catch is that NET

needs access to the handover data of a particular subscriber in order to pinpoint their location geographically. Like the GSM protocol itself, the idea of Cell Probe is fairly straightforward. NET claims that they can, either by doing field research or by calculating based on antenna power, determine where the borders of each cell physically lie. That is, they can know the location of all of the points at which someone would cross over from BTS_1 to BTS_2 from the previous chapter, for example, where a handover would be initiated. Say, then, that a particular subscriber was handed over from BTS_1 to BTS_2 , and this handover is reflected in the VLR and the GMSC. Then the NET software would be able to consider all of the physical roadways that go from cell 1 to cell 2, since the coordinates of all roadways are already known. By repeating this procedure per handover, NET claims that they can (1) determine the exact path on which any particular subscriber is driving, and (2) calculate the distribution of trip times for all people traveling on that road. A simple form of the Cell Probe concept is diagrammed below:

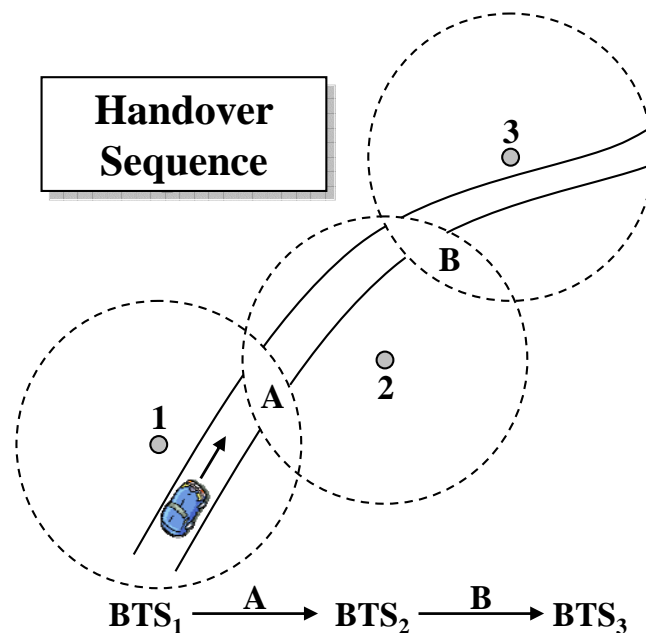


Figure 3.1. Simple handover sequence with two handovers performed.

The previous diagram displays the key concept behind Cell Probe. While the cellular user drives her car along the road, the handover from BTS_1 to BTS_2 is recorded (label A). Then, farther down the road, the user crosses from BTS_2 coverage into BTS_3

coverage and the handover is again recorded (label B). Since the locations of A and B are already known, a filtering procedure can be applied to narrow down the list of different paths on which the user could be traveling. Obviously this calculation is trivial in the diagram on the previous page, since only one path exists. NET, however, is claiming that this procedure can be performed in complex urban environments with tens or even hundreds of underlying paths. Some slides from their road-show presentation detailing this claim are laid out below:

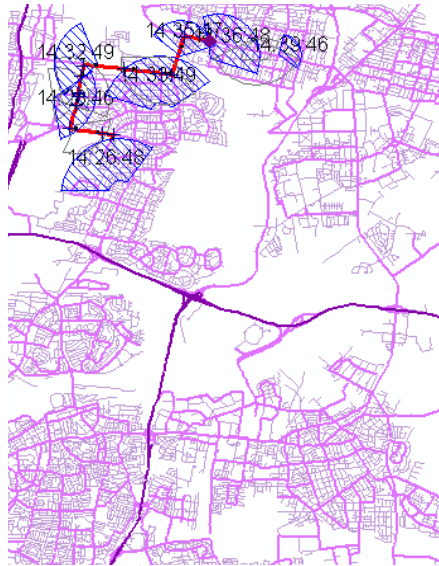


Figure 3.2

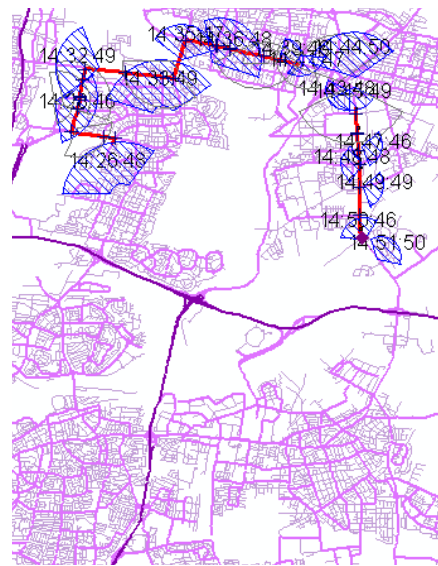


Figure 3.3

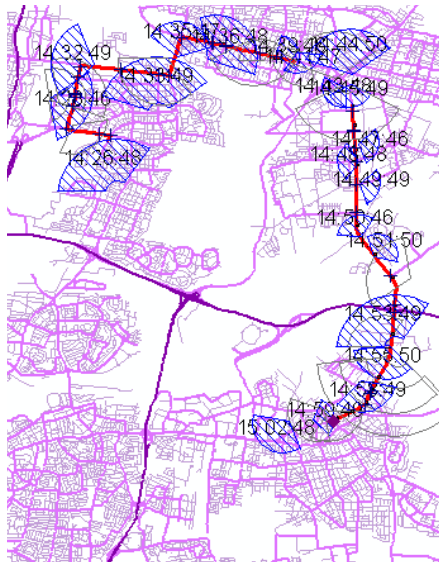


Figure 3.4

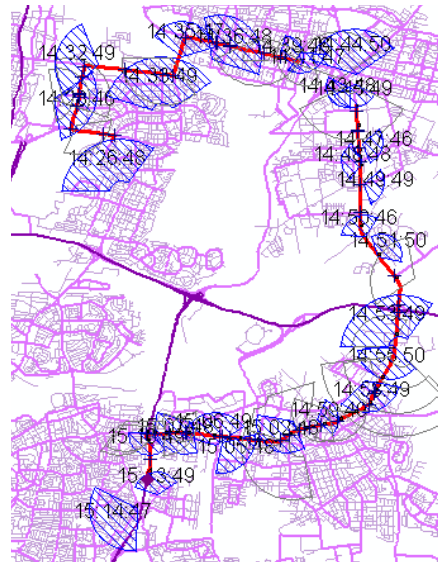


Figure 3.5

The slides from NET's presentation, progressing from Figure 3.2 to Figure 3.5, show a user driving along a specific path while being tracked using cellular handover and the NET software. Notice that the areas of cellular coverage, shown shaded in blue, are not full circles and indicate that 120-degree directional antennas are used in the area. Also note the outer edges of each antenna's signal ends discretely in NET's slides. In other words, the NET slides imply that, in the absence of another BTS to distribute up the signal from a traveling user who is traveling away from their active BTS, the BTS that is actively servicing the user would stop cellular service at some fixed distance away.

In the process of writing this paper, all research has indicated that the implication that a BTS signal can be calculated deterministically at a given distance, in reality, is physically impossible. First, any radio signal being broadcast from a fixed point is subject to two stochastic processes: multipath and fading (Schwartz, Bennett, and Stein 347). Every radio signal experiences multipath, or the statistic probability of taking a distribution of different paths from origin to destination, due to both physical natural and man-made obstacles. These same signals have been empirically shown to undergo Rayleigh fading, meaning that they fade randomly according to the Rayleigh distribution (Ibid 348). More specifically, Rayleigh fading is defined as

$$f_R(r|\sigma) = \frac{r}{\sigma^2} e^{-r^2/2\sigma^2}, \quad r \geq 0 \quad (3.1)$$

with moment-generating function

$$M(t) = 1 + \sigma t \exp\left(\frac{\sigma^2 t^2}{2}\right) \sqrt{\frac{\pi}{2}} \left(\frac{2}{\sqrt{\pi}} \int_0^{\sigma t/\sqrt{2}} e^{-t^2} dt \right) - 1 \quad (3.2)$$

where R is the power at a distance r from the broadcast antenna with some calibrated standard deviation σ . It should be noted that wireless signal has only been empirically shown to exhibit Rayleigh fading. That is, the Rayleigh fading equation is only an approximation of signal strength at a given distance. The exact equation for signal

strength distribution at time t , when all multipaths and fading have been considered is (Schwartz, Bennett, and Stein 357)

$$s(t) = \text{Re} \left\{ \exp(j2\pi f_0(t-t_0)) \int_{-\infty}^{\infty} \beta(\tau; t-t_0) \mu_0(t-t_0-\tau) d\tau \right\} \quad (3.3)$$

with “frequency-time cross-variance” (Ibid 360)

$$R_T(\tau_1, \tau_2; \Delta t) = \frac{1}{2} \langle \beta^*(\tau_1; t) \beta(\tau_2; t + \Delta t) \rangle = r_T(\tau_1; \Delta t) \delta(\tau_1 - \tau_2) \quad (3.4)$$

and (Ibid 362)

$$R_F(f_1, f_2; \Delta t) = \int_{-\infty}^{\infty} r_T(\tau; \Delta t) \exp(j2\pi(f_1 - f_2)\tau) dt \quad (3.5)$$

The exact signal strength equations listed above are taken from a book designed to assist PhD candidates in complex signal research. The constants j and u_0 and Weiner process $\beta(\tau; t-t_0)$ alone are far too intricate and complicated for the scope of this paper, so their treatment will end here and the Raleigh approximation will be assumed. Needless to say, NET’s implied assumption that signal strength can be deterministically calculated at set distances from the BST is an invalid one since the signal strength approximation itself is a stochastic process.

NET’s overall claim is a sizable one. Clearly, knowing the exact path on which a subscriber was driving would allow people to replace their expensive GPS units with common cell phones, and knowing the distribution of travel times along that path would bring about near-perfect traffic analysis and trip-planning. But is such a claim feasible? What are the conditions under which such a system would operate correctly and produce meaningful results? Under which conditions would it fail?

The Cell Probe technology itself is private, so the exact algorithms and calculations required to accurately position cellular clients are unknown. This paper will serve as a “best guess” effort to understand the possible steps required in identifying a user’s location based only on cellular handover. In addition, the paper will go beyond the

NET Cell Probe framework and discuss other possible uses of handover data such as BTS location searching. Finally, the paper will describe and disclose in detail a functional model that emulates a working system of roads to test different scenarios under which Cell Probe technology is or is not useful.

It was shown earlier in the chapter that BTS signal to a GSM handset does not discretely end at some fixed distance. Therefore, this paper will disregard any concept that implies that path-location can result from a user traveling outside a BTSs range. As a result, only handovers between two explicit BTSs will be considered, as this is much more realistic.

Exploring the Claim

The following will describe the process by which proper path selection can result from only coverage handover data. Again, the Cell Probe path selection routine is private, so this paper will illustrate a proprietary routine called `PathLocator`, included at the end of this thesis under *Appendix A*. While it cannot be shown with certainty that `PathLocator` performs similar calculations and sorts those of Cell Probe, it can be assumed with a fairly high level of confidence that the steps of both routines are strikingly similar if not identical. In essence, `PathLocator` will take the real-life handover data from a given subscriber traveling on an unknown path, defined here as $\xi(T)$, and compare that data with the generated handover data from all known paths. If the data in $\xi(T)$ matches the data in one or more of the known paths, then the “feasible set of paths” $\varphi(P)$ takes the values of those known paths. In other words, the traveling subscriber is most likely on one of the paths listed in $\varphi(P)$ at the end of the routine.

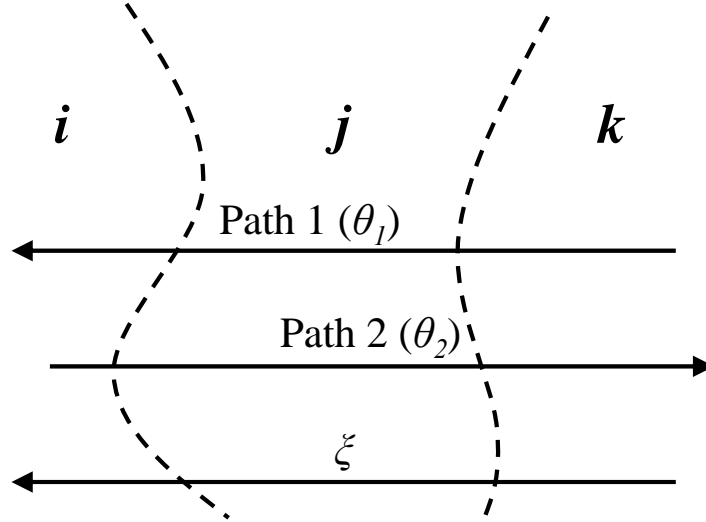


Figure 3.6. Two simple paths with two handovers each and one unknown path.

Consider the figure above that shows two sample paths, perhaps taken to represent the two opposite lanes of a highway. In this particular scenario, defined as the state space Ω , The BTSs servicing the area are labeled i , j , and k , with path θ_1 spanning from i to j to i , and path θ_2 spanning from i to j to k . Before any other calculations are made, PathLocator must read into memory all possible BTSs from some database, building its reference set $\psi(T)$. Mathematically,

$$\psi(T) = \{t_i : \exists t_i, i = 1 \dots M\} \quad (3.6)$$

where M is some arbitrarily large number. At this point, the reference set $\psi(T)$ is the set containing all of the BTSs in existence regardless of their status. PathLocator then needs to take under consideration the status of each BTS and ignore it as active if it is under construction, under maintenance, or simply not broadcasting:

$$\psi(T) = \{\psi(t_i) : t_i \in A_T, i = 1 \dots |\psi(T)|\} \quad (3.7)$$

where A_T is the set of currently active towers. Note that $\psi(T)$ is defined recursively in (3.7) and the original reference set is overwritten. In this case,

$$\psi(T) = \{i, j, k\} \quad (3.8)$$

Now that PathLocator knows the set of all operational towers $\psi(T)$, it can consider the unknown path $\xi(T)$. In order to do so, first PathLocator needs to read into memory the handover data for $\xi(T)$. It is important to preserve the direction in which the user has traveled along $\xi(T)$, so strict ordered sets will be used in favor of weak or reflexive naïve sets. These ordered sets will represent the directed arcs that are each path. Define the variable

$$C_{\text{adjacent}}(\psi(t_1), \psi(t_2)) = \begin{cases} 1, & \text{if } t_1 \text{ is connected to } t_2 \\ 0, & \text{if } t_1 \text{ is not connected to } t_2 \end{cases} \quad (3.9)$$

as an index variable where two regions are connected if a user can travel from one directly to the other if an infinite number of paths existed. For example, under this scenario,

$$\begin{aligned} C_{\text{adjacent}}(i, j) &= C_{\text{adjacent}}(j, i) = 1 \\ C_{\text{adjacent}}(i, k) &= C_{\text{adjacent}}(k, i) = 0 \\ C_{\text{adjacent}}(j, k) &= C_{\text{adjacent}}(k, j) = 1 \end{aligned} \quad (3.10)$$

Let

$$\omega = \{(m, n) : m \neq n, m \in \psi(T), n \in \psi(T), C_{\text{adjacent}}(m, n) = 1\} \quad (3.11)$$

be the set of all possible strictly ordered pairs representing handover under Ω with $C_{\text{adjacent}} = 1$. In this particular case, in the diagram above,

$$\omega = \{(i, j), (j, i), (j, k), (k, j)\} \quad (3.12)$$

while it is important to note that the directed arcs $(i, j) \neq (j, i)$ under Ω .

Next, PathLocator defines the placeholder variable

$$\alpha = \{\omega_m : m \text{ is the first handover sequence}\} \quad (3.13)$$

or the first ordered pair of towers that provided coverage to the subscriber. In this case,

$$\alpha = (k, j) = \omega_m, m = 4 \quad (3.14)$$

With all of the relevant variables initialized, the unknown path $\xi(T)$ can be described as

$$\xi(T) = (\alpha_1, \alpha_2, \psi(t_1), \dots, \psi(t_n)), n = 1 \dots (N - 2) \quad (3.15)$$

where N is the total number of different towers that provide coverage along path $\xi(T)$ under Ω . In this example,

$$\xi(T) = (k, j, i) \quad (3.16)$$

At this point, the feasible set $\varphi(P)$ is initialized as well:

$$\varphi(P) = \left\{ \begin{array}{l} (m_1, m_2, \dots, m_n) : m_i \neq m_{(i+1)}, m \in \psi(T), \\ C_{\text{adjacent}}(m_i, m_{(i+1)}) = 1, n = 1 \dots N, \forall i \end{array} \right\} \quad (3.17)$$

Under the current example,

$$\varphi(P) = \{(i, j, k), (i, j, i), (j, i, j), (j, k, j), (k, j, i), (k, j, k)\} \quad (3.18)$$

Now PathLocator needs to determine the order of BTS coverage for each other path, defined as the set $\Theta(T)$, under Ω to compare to $\xi(T)$, given the set of all operational towers $\psi(T)$ and the set of all possible handovers ω . By comparing the other paths to $\xi(T)$, the feasible set $\varphi(P)$ can be pared down significantly until it contains only the most likely candidates for $\xi(T)$. The other paths $\theta(T)$ under Ω are defined in a similar fashion to that in which $\xi(T)$ was defined. In the spirit of brevity, assume that each other path has been defined in the same manner as $\xi(T)$ and is stored in $\theta(T)$. Then, in this case

$$\theta_1(T) = (k, j, i) \quad (3.19)$$

and

$$\theta_2(T) = (i, j, k) \quad (3.20)$$

Now that all relevant variables have been defined, PathLocator can call FilterPaths, the subroutine that compares the unknown path $\xi(T)$ to the possible paths $\varphi(P)$ while reducing $\varphi(P)$ as much as possible.

Procedure FilterPaths($\xi(T)$, $\varphi(P)$, $\theta(T)$)

Step 1. For all $\theta_i \in \Theta$:

Step 1a. Initialize: Set $\delta = \text{True}$ % Boolean variable to keep searching

Step 2. For all $\varphi_i \in \varphi$: % Remove paths that don't exist

Step 3. Do while $\delta = \text{True}$

Step 4. If $\Theta \supset \varphi_i$ Then

Step 5. Continue

Else:

Step 5a. Remove φ_i from φ

Step 6. If $\varphi = \emptyset$ Then

Set $\delta = \text{False}$

Step 7. For $n = 1$ to $|\xi|$:

Step 8. For all $\varphi_i \in \varphi$:

Step 9. If $\xi_n = (\text{element } n \text{ of } \varphi_i)$

Step 10. Continue

Else:

Step 10a. Remove φ_i from φ

End Procedure

The sorting and paring of the feasible set $\varphi(P)$ are listed above procedurally. Note that passing the variables $\xi(T)$, $\varphi(P)$, and $\theta(T)$ to `FilterPaths` is valid at this point as they have all been properly defined. After the `FilterPaths` procedure runs, the only elements φ_i of φ that have not been pared out are the ones that match exactly the n -tuple defined in $\xi(T)$. Now one last procedure, `StoreMatches`, determines which explicit path produces a match in $\xi(T)$ and $\varphi(P)$ and ultimately stores them in the index variable

$$\gamma = \{(B_1, \dots, B_n) : B \in \{0,1\}, n = 1 \dots |\Theta(T)|\} \quad (3.21)$$

where $|\Theta(T)|$ is the number of paths other than $\xi(T)$, or the length of $\Theta(T)$.

`StoreMatches`, like `FilterPaths`, is defined procedurally on the next page.

Procedure StoreMatches ()

Step 1. For all $\theta_i \in \Theta$:

Step 2. For $n = 1$ to $|\gamma|$: % Initialize γ

Step 3. Set $|\gamma_n| = 1$

Step 4. If $\theta_i \in \varphi$ Then:

Step 5. Continue

Else:

Step 5a. Set $\gamma_i = 0$

End Procedure

Finally, after all of the subroutines have been performed, `PathLocator` produces the final vector γ , indicating which path θ_i (if any) matches the subscriber's path $\xi(T)$.

To prevent confusion, it should be noted that γ is an ordered set like the rest of the path

variables defined for PathLocator, though it does not represent some directed arc as do $\xi(T)$, $\varphi(P)$, and $\theta(T)$. In the current example,

$$\gamma = (1, 0) \quad (3.22)$$

indicating that θ_1 does, indeed, match $\xi(T)$ since $\gamma_1 = 1$ while θ_2 does not match $\xi(T)$ since $\gamma_2 = 0$. The results indicate that there exists only one exact match to $\xi(T)$, and in this case it is correct since

$$\xi(T) = (k, j, i) = \{\theta_i : \gamma_i = 1\} = \theta_1 \quad (3.23)$$

In this instance of the state space Ω the PathLocator procedure has detected the user's correct path. It should be reiterated that the algorithm that powers Cell Probe is most likely, although not necessarily exactly similar to PathLocator. In any case, is the PathLocator algorithm robust, or did it only select correctly in the previous example as a result of perfect conditions? With respect to the algorithm's overall power, what does a closer look into other possible scenarios expose about the algorithm's reliability?

The Algorithm is Weak

Consider instead the following scenario diagrammed below:

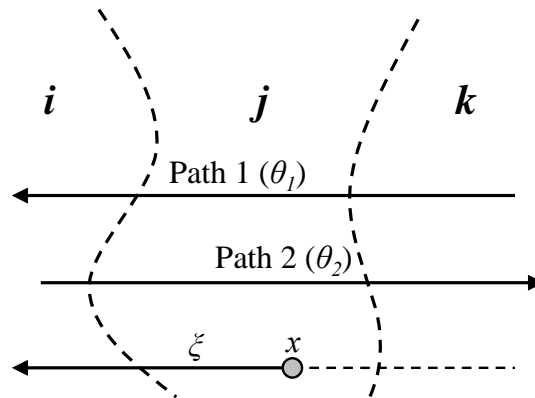


Figure 3.7. Sample path with missed handover data.

This state space Ω looks similar to the previous space, though the subscriber's first handover ω_1 in $\xi(T)$ has been missed due to some unknown influence in the

scenario. Perhaps, though the user was indeed driving along θ_1 , her phone was turned off while she was in region k at point x . Examining the behavior of PathLocator under these circumstances reveals some disturbing results. First, $\psi(T)$ and ω are both defined correctly:

$$\psi(T) = \{i, j, k\} \quad (3.24)$$

and

$$\omega = \{(i, j), (j, i), (j, k), (k, j)\} \quad (3.25)$$

but α , and subsequently $\xi(T)$, are defined differently:

$$\alpha = \{\omega_m : m \text{ is the first handover sequence}\} = (j, i) = \omega_m, m = 2 \quad (3.26)$$

and

$$\xi(T) = (\alpha_1, \alpha_2) = (j, i), N = 2 \quad (3.27)$$

Notice that since $|\xi(T)| = N = 2$ now, $\varphi(P)$ will be initialized incorrectly:

$$\begin{aligned} \varphi(P) &= \{(m_1, m_2) : m_1 \neq m_2, m \in \psi(T), C_{\text{adjacent}}(m_1, m_2) = 1, N = 2\} \\ &= \{(i, j)(j, i)(j, k)(k, j)\} \end{aligned} \quad (3.28)$$

Remember that both θ_1 and θ_2 contain 2 handovers, thereby requiring at a set length of at least 3. Iterating over the rest of the procedure is trivial at this point, because

$$(|\theta_m| = 3 > 2 = |\varphi_n|) \quad \forall m \quad \forall n \quad (3.29)$$

Since no possible path in $\varphi(P)$ is the same length as any path in $\Theta(T)$, then at the end of the procedure

$$\varphi(P) = \emptyset \quad (3.30)$$

and

$$\gamma = (0, 0) \quad (3.31)$$

This implies that the subscriber was not traveling on any of the paths in the state space Ω . Clearly this is incorrect, as the user is traveling on some path in $\Theta(T)$, since $\Theta(T)$ explicitly defines all of the paths in Ω . This demonstrates the extremely high sensitivity that PathLocator exhibits. Simply missing 1 subscriber handover leads to a feasible set of zero! Given that, while driving, an individual handset will be subject to tens or even hundreds of handovers, this behavior of the algorithm is unacceptable.

Several different scenarios would cause PathLocator to produce as its match the empty set. Consider the following state space Ω diagrammed on the below:

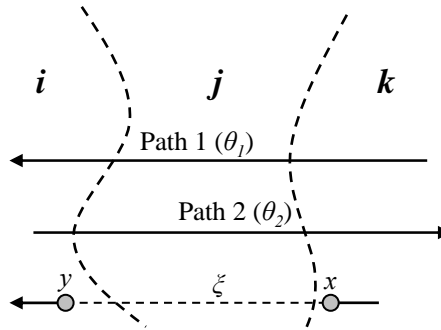


Figure 3.8. Sample path with missed handover data.

In this case, perhaps BTS_j was under routine maintenance and was not broadcasting signal while the subscriber was driving along $\xi(T)$. In effect, the coverage of BTS_k would have been enough to continue service from x until the subscriber reached the coverage of BTS_i at point y . It would be redundant to show that this situation would again cause PathLocator to produce no matches, though it does do well to illustrate the point.

It has been shown that PathLocator is unreliable when as few as 1 handover is missed. A simple Monte Carlo analysis produces insight as to the effects of this behavior. Refer to *Appendix A* for the code used to generate the following plot.

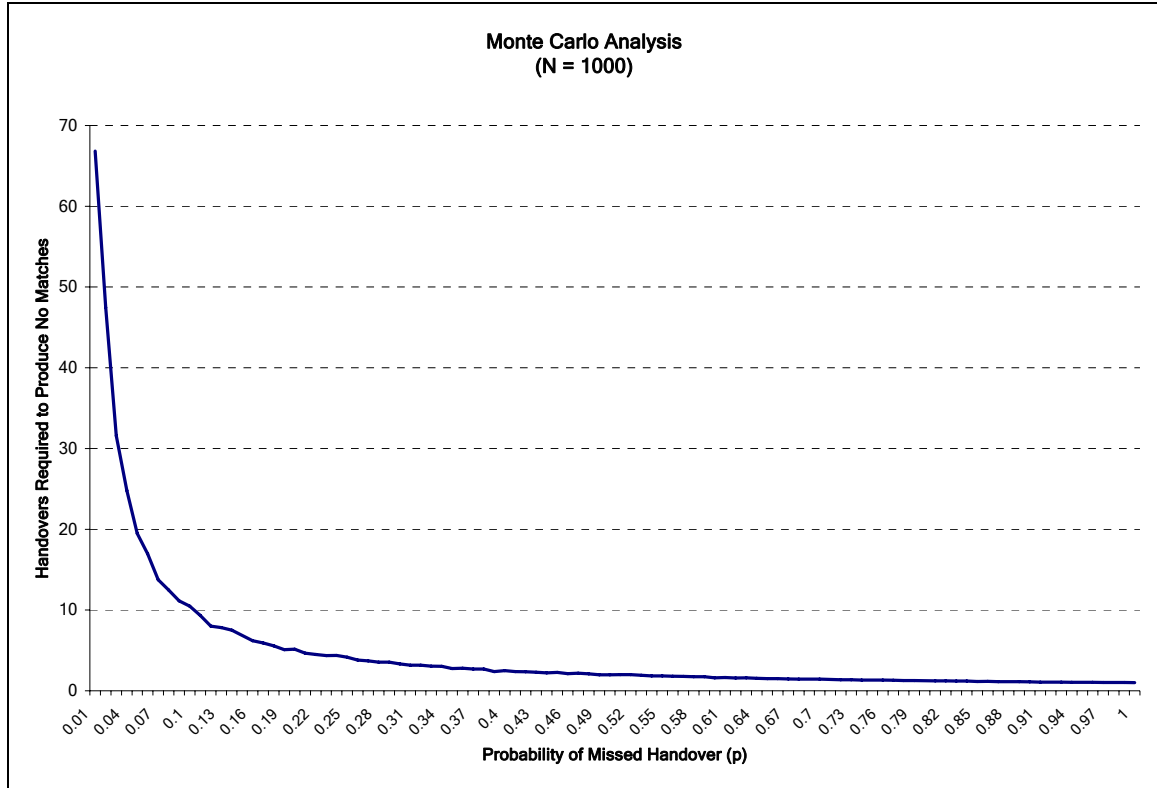


Figure 3.9. Monte Carlo analysis of PathLocator behavior.

The graph above compares the number of handovers required for PathLocator to return \emptyset as a function of the probability of a missed handover p . Notice that a value as low as $p = 0.05$ requires only 19 handovers for PathLocator to fail. Considering the fact that NET’s Cell Probe slides show more than 19 handovers occurring on their user’s trip, the need to modify the PathLocator algorithm in some way becomes evident.

A Better Approach

PathLocator can be improved to include something that this paper will refer to as “soft searching”, in contrast with the “hard searching” originally performed. PathLocator’s hard searching immediately returned the empty set when an exact match of $\xi(T)$ was not found. By contrast, soft searching will return a path match if the

handover behavior exhibited in $\xi(T)$ is found anywhere along one of the paths located in $\Theta(T)$. Let this routine be called `SoftSearch`. The goal of `SoftSearch` searching is

$$\gamma_i = \left\{ B_i = \begin{cases} 1, & \text{if } \theta_i \subseteq \xi(T) \\ 0, & \text{otherwise} \end{cases} \right\} \quad (3.32)$$

Consider the scenario Ω diagrammed below.

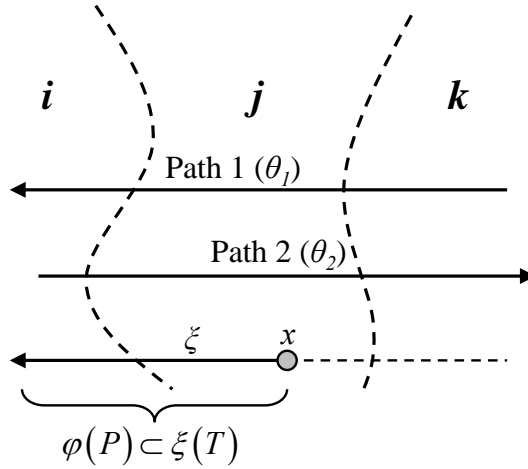


Figure 3.10. Sample path with missed handover data.

In order to allow for soft searching, the algorithm must define the initial variables in a different way. The initial aim of `SoftSearch` is to initialize $\varphi(P)$ so that

$$\varphi(P) \subseteq \xi(T) \quad (3.33)$$

This can be achieved by initializing $\varphi(P)$ in a different manner than that of `PathLocator`. In order to do so, however, the initialization of $\varphi(P)$ needs to be done procedurally. Note that $|\theta(t_{\max})|$ is the length of the longest path (having the most handovers) in $\Theta(T)$.

Procedure InitializePhi($\psi(T), \theta(T)$)

Step 0. Initialize: $\varphi = \emptyset$

Step 1. For $n = 1$ to $\lceil \theta(t_{\max}) \rceil$:

Step 2. If $n = 1$ Then

Step 3. Add $\psi(t_n)$ to φ

Else

Step 4. Add φ' to φ , where $\varphi' = \left\{ \begin{array}{l} (m_1, \dots, m_r) : m_i \neq m_{(i+1)}, m \in \psi(T), \\ C_{\text{adjacent}}(m_i, m_{(i+1)}) = 1, r = 1 \dots n, \forall i \end{array} \right\}$

End Procedure

Since SoftSearch initializes the feasible set $\varphi(P)$ differently than does

PathLocator, InitializePhi produces

$$\varphi(P) = \{(i), (j), (k), (i, j), (j, k), (k, j), (k, i), (i, j, k), (k, j, i)\} \quad (3.34)$$

under the current scenario. SoftSearch requires only one more variable

$$\phi_{i,n} = \{a : a \text{ is the } n^{\text{th}} \text{ element of } \varphi_i\} \quad (3.35)$$

Now, SoftSearch proceeds in the same fashion as PathLocator, except with a newly improved set of variables. It executes the SoftFilterPaths procedure defined on the next page to pare down $\varphi(P)$, but is much more conservative than PathLocator in removing φ_i from φ .

Procedure SoftFilterPaths($\xi(T), \varphi(P)$)

Step 1. For all $\varphi_i \in \varphi$:

Step 1a. Initialize: $\Delta = 0$ % Offset variable

Step 1b. Initialize: $k = 1$ % Index variable

Step 1c. Initialize: $\delta_{found} = \text{False}$ % Has a match been found?

Step 2. If $|\varphi_i| < |\xi|$

Step 3. Remove φ_i from φ % Remove paths shorter than ξ

Else:

Step 4. Do while $(\Delta < |\varphi_i| - |\xi| + 1)$

Step 5. If $\xi_k = \phi_{i,(\Delta+k)}$

Step 6. $k = (k + 1)$ % Increment k

Step 7. If $k = (|\xi| + 1)$

Step 8. $\delta_{found} = \text{True}$

Step 8a. $\Delta = (|\varphi_i| - |\xi| + 1)$

Else:

Step 9. $\Delta = (\Delta + 1)$ %Increment Δ

Step 9a. $k = 1$

Step 10. If $\delta_{found} = \text{False}$

Step 11. Remove φ_i from φ

End Procedure

SoftSearch can use the same procedure as PathLocator to store the path-matching results of SoftFilterPaths. After running StoreMatches, the end result of using SoftSearch is much more desirable than that of using PathLocator under the same scenario:

$$\varphi(P) = \{(j, i)\} \quad (3.36)$$

and

$$\gamma = (1,0) \quad (3.37)$$

which is the correct result.

In order to produce correct results when some handovers are missed, SoftSearch must make some sacrifices relative to PathLocator. In the case of PathLocator, a path match implies with complete accuracy that the correct path has been found. This occurs because PathLocator will return 1 and only 1 result if it returns any at all. By contrast, SoftSearch returns path matches of any path that contains the unknown path. As a result, it could return up to all of the possible other paths under the worst case scenario.

Consider the scenario diagrammed below:

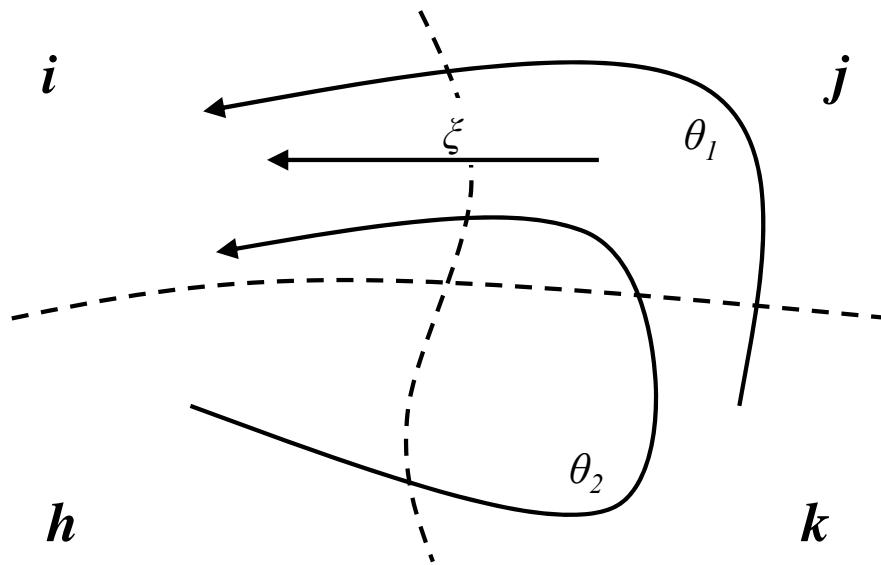


Figure 3.11. Worst-case scenario for SoftSearch.

In this scenario, running SoftSearch would provide no real insight as to which path the subscriber was using to travel. First, $\varphi(P)$ would be initialized as

$$\varphi(P) = \left\{ \begin{array}{l} (h), (i), (j), (k), (h, i), (i, h), (i, j), (j, i), (j, k), \\ (k, j), (k, h), (h, k), (h, i, j), (j, i, h), (i, j, k), \\ (k, j, i), (j, k, h), (h, k, j), (k, h, i), (i, h, k), \\ (h, i, j, k), (k, j, i, h), (i, j, k, h), (h, k, j, i), \\ (j, k, h, i), (i, h, k, j), (k, h, i, j), (j, i, h, k) \end{array} \right\} \quad (3.38)$$

After running `SoftFilterPaths` and `StoreMatches`, the resulting variables would be

$$\varphi(P) = \{(j, i), (j, i, h), (k, j, i), (k, j, i, h), (h, k, j, i), (j, i, h, k)\} \quad (3.39)$$

and

$$\gamma = (1, 1) \quad (3.40)$$

Clearly this is incorrect as no user can travel on two roads at once! The effectiveness of `PathLocator` and `SoftSearch` will be explored in the next chapter.

Finding the Towers

NET claims that knowing handover data in a given physical area provides enough information to match paths to traveling subscribers. This claim assumes that the physical location of the points at which a handover would occur is known. Let

$$L(\omega_i) = (x, y) \quad (3.41)$$

represent the physical x and y Cartesian coordinates of a crossover ω_i . Clearly, Cartesian coordinates are not used in the real world due to several factors, but their treatment for the purposes of this paper will be sufficient.

Since, in reality, the exact coordinates of a handover ω_i are (1) not known and (2) subject to change as a result of signal interference, Rayleigh fading, and a number of other factors, define the random variable

$$L_{\text{estimate}} = (x + \varepsilon, y + \varepsilon) \quad (3.42)$$

where,

$$\varepsilon \sim N(\mu, \sigma^2) \quad (3.43)$$

and μ and σ are some calibrated mean and standard deviation.

Now a procedure named TowerLocator can be constructed to approximate the physical coordinates of a given BTS based on the handovers into its coverage zone and the handovers out of its coverage zone. Let

$$\beta_{in}(i, n) = \left\{ (x_i, y_i) : \begin{cases} x_i \text{ is the } x \text{ coordinate of the } n^{\text{th}} \text{ handover into region } i \\ y_i \text{ is the } y \text{ coordinate of the } n^{\text{th}} \text{ handover into region } i \end{cases} \right\} \quad (3.44)$$

and

$$\beta_{out}(i, n) = \left\{ (x_i, y_i) : \begin{cases} x_i \text{ is the } x \text{ coordinate of the } n^{\text{th}} \text{ handover out of region } i \\ y_i \text{ is the } y \text{ coordinate of the } n^{\text{th}} \text{ handover out of region } i \end{cases} \right\} \quad (3.45)$$

be the variables that store the Cartesian coordinates for the handovers into and out of region i . Let

$$\lambda_x(i, n, d) = \begin{cases} \text{the } x \text{ coordinate of } \beta_{in}(i, n), & \text{if } d = 1 \\ \text{the } x \text{ coordinate of } \beta_{out}(i, n), & \text{if } d = 2 \end{cases} \quad (3.46)$$

and

$$\lambda_y(i, n, d) = \begin{cases} \text{the } y \text{ coordinate of } \beta_{in}(i, n), & \text{if } d = 1 \\ \text{the } y \text{ coordinate of } \beta_{out}(i, n), & \text{if } d = 2 \end{cases} \quad (3.47)$$

Then TowerLocator can approximate the actual coordinates of BTS_i by calculating

$$x_{\min}(i) = \arg \min_{x_0} \left(\sum_n (\lambda_x(i, n, 1) - x_0)^2 + (\lambda_x(i, n, 2) - x_0)^2 \right) \quad (3.48)$$

and

$$y_{\min}(i) = \arg \min_{y_0} \left(\sum_n (\lambda_y(i, n, 1) - y_0)^2 + (\lambda_y(i, n, 2) - y_0)^2 \right) \quad (3.49)$$

The best approximation for the physical location of BTS_i is then

$$L_{\text{estimate}} = (x_{\min}(i), y_{\min}(i)) \quad (3.50)$$

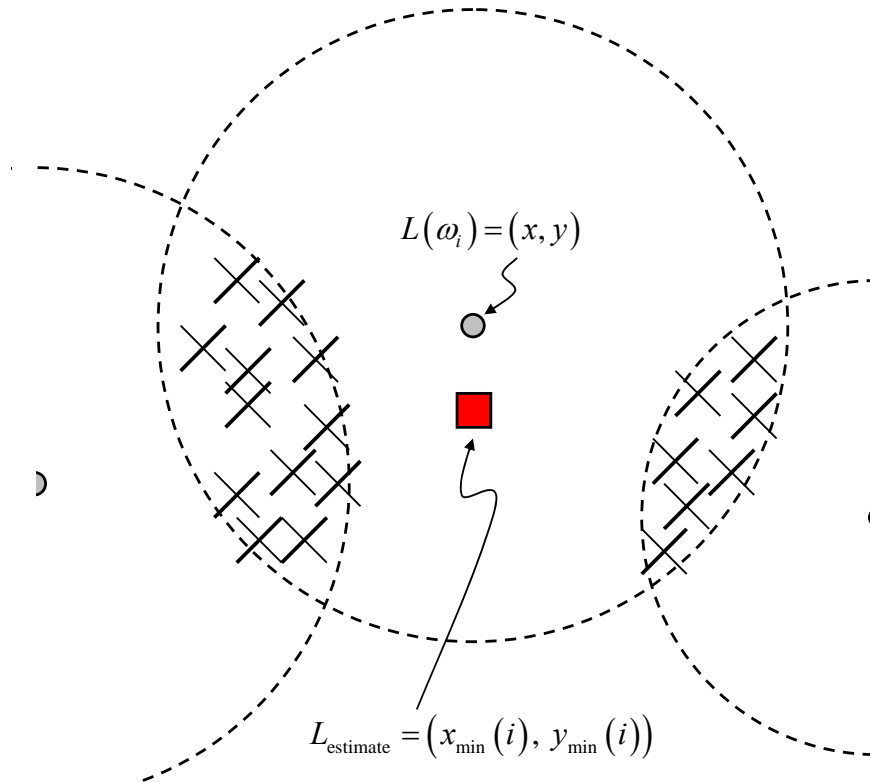


Figure 3.12. Example of TowerLocator.

The figure above is an example of how TowerLocator would fare in a given scenario. The large black \times marks indicate locations on which handovers occurred. The red square in the center of the chart is TowerLocator's best guess to the actual BTS_i location (the gray circle at the center of the diagram). Notice that the error-minimizing coordinates are more accurate in the horizontal direction than in the vertical direction. This occurs because the handovers in this case encapsulate the horizontal component of the tower's coordinates (the handovers occur both to the left and to the right of the tower) but do not encapsulate the vertical component (the handovers mostly occur below the tower's coordinates).

Though one might think TowerLocator to seem trivial, it is important to highlight the procedure's inherent value do dispel such myths. According to NET's claims, the physical location of each handover is known as the requirement for path matching under Cell Probe. If this information is already known, then TowerLocator can be written for some computer system at no extra cost. And although the procedure might not preclude the need for field testing and maintenance, it might speed up some process of organization and categorization of cellular BTSs at some niche in the huge telecom industry. By doing this TowerLocator would effectively lower costs some amount, even if that amount were relatively small. If this were to be the case, then TowerLocator would prove not to be trivial after all.

“I’m no model lady. A model is just an imitation of the real thing.”

--Mae West

4 Building a Model

At the time of this writing, some of the major telecommunications giants include Verizon/MCI, SBC/AT&T/Cingular, and Sprint/Nextel, though these companies are constantly subject to additional acquisitions or divestments that may cause them to no longer exist in a few years time. Given the fact that these companies are being revalued on a nearly real-time basis, the telecoms themselves act with a hyper-sensitivity to the public’s perception of their fiscal responsibility and subscriber privacy policies. A negative story in the press about unsecured user data involving one of these companies could result in a drop of millions of dollars in market capitalization and significantly impair the ability of the company to compete in today’s chaotic telecom mergers and acquisitions (M&A) race.

With this in mind, it becomes apparent that searching for real-life handover data would pose a significant problem for several reasons. First and foremost, telecom

companies are extremely reluctant to disclose any sort of subscriber information whatsoever to protect themselves against privacy lawsuits and scandals. In this context, they would only provide user data – IMSI numbers, handover data stored in the BSCs, etc. – upon court subpoena, as this writer was informed upon contacting several of the larger companies. Furthermore, even if the management of such a telecom were inclined to release this information, the sheer breadth of their networks and the volumes of data contained therein would complicate significantly the search for specific and usable handover data. In addition, the constant M&A activity within the telecom sector effectively causes M&A activity of different network structures and protocols within the companies themselves. For example, before their merger, Sprint used the CDMA protocol while Nextel used a GSM standard (Bernatchez). The merger of the two companies forced Nextel to convert all of its existing network hardware to become CDMA compliant and purchase new equipment where necessary. Searching for handover data in such a situation would be extremely tedious as the protocols themselves are not well defined within such a company. For reference, SBC/AT&T/Cingular uses a GSM protocol while Verizon/MCI operates under the CDMA standard (Ibid). In the face of these complications, how then can one construct a proper model that tests the effectiveness of PathLocator, SoftSearch, and TowerLocator without proper handover data? Without a tool to test these routines firsthand, how can a reliable estimate Cell Probe's feasibility be obtained? The answer lies in creating working model that effectively creates handover data itself for a given state space Ω through simulation.

Generating the Data

To facilitate the testing of the procedures described in chapter 3, a working model was developed with several objectives in mind. The exact code and screenshots of the

spreadsheets used can be found in *Appendix A*. The model was required to satisfy several requirements and was constructed with these in mind. Ultimately, the model needed to:

- (1) Contain a map of a virtual state space Ω where the engineer using the model could draw paths and towers that would represent real-world roads and BTSs.
- (2) Simulate cellular subscribers traveling along the paths drawn in (1).
- (3) Record the handover data that resulted from the traveling subscribers in (2) and the drawn towers in (1).
- (4) Run PathLocator and SoftSearch to estimate the path of each subscriber and compare the results to the true path of the subscriber.
- (5) Run TowerLocator to approximate the location of a tower drawn in (1).

Microsoft Excel and its scripting language, Microsoft Visual Basic (VB), were chosen as the platform and programming language used to build the model described above and the model itself was named SimulateSubscriber. It should be noted that Microsoft Visual Basic is different than Microsoft Visual Basic.NET, which allows for object-oriented programming, creation of templates, and other advanced methods that are not included in Microsoft Visual Basic. This chapter will serve to describe in detail the construction of SimulateSubscriber, highlight the more important calculations required for the model to function, and compare the theoretical procedures listed in chapter 3 with the VB code used to implement them.

Drawing the World

To create a realistic system of paths and towers, perhaps a good place in which to begin is with the mapping system Google Maps (<http://maps.google.com>) and the online FCC database of currently active/inactive cellular towers

(<http://wireless.fcc.gov/antenna>). Cross referencing the towers listed in the FCC database with the maps generated by Google Maps would create a good foundation from which to build a state space Ω in SimulateSubscriber. A search of Princeton University and its surrounding areas brings up the following in Google Maps:

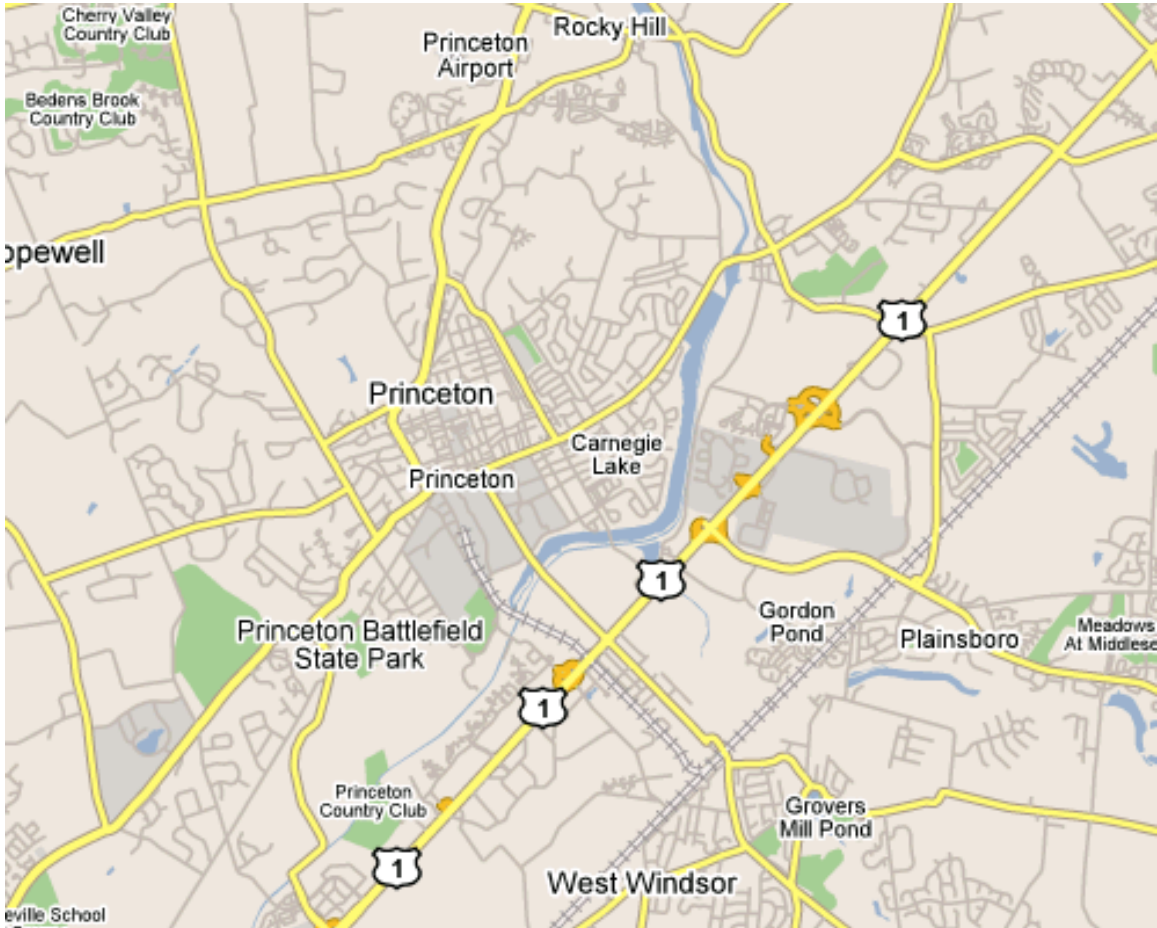


Figure 4.1. Google Maps representation of Princeton University and surrounding Township.

A search of the online FCC database of cellular towers and structures around Princeton University returns:

| Registration Number | Status | File Number | Owner Name | Latitude/Longitude |
|-------------------------|-------------|-------------|---|-----------------------------|
| 1001881 | Constructed | A0002050 | TKR CABLE COMPANY | 40-12-16.0N 074-39-51.0W |
| 1003347 | Cancelled | A0003800 | AT&T WIRELESS SERVICES INC | 39-56-26.0N 074-58-49.0W |
| 1004050 | Terminated | A0004722 | AT&T WIRELESS SERVICES INC | 39-49-36.0N 075-09-26.0W |
| 1005215 | Constructed | A0485448 | Spectrasite Communications, Inc. through American Tower, Inc. | 40-15-03.2N 074-15-06.2W |
| 1006539 | Constructed | A0465049 | NEW CINGULAR WIRELESS SERVICES, INC. | 40-37-50.0N |

Figure 4.2. FCC database snapshot of cellular structures around Princeton University.

Fortunately, Google Maps is generous enough to distribute code libraries that allow the placement of custom objects within the maps produced. Because of this, one could write an applet that reads the structure location information from the FCC and paints tower objects onto a figure generated by Google Maps accordingly. Mobicedia (<http://www.mobicedia.com>), a small internet company founded in 2002 to track cellular coverage, has done just that. A proprietary PHP script that Mobicedia has built brings up the following search box:

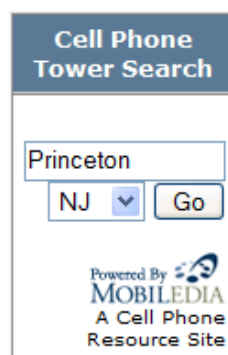


Figure 4.2. Mobicedia applet searching for Princeton, NJ.

The code used to create the search box in Figure 4.2 is included in *Appendix A*. After receiving a search request for Princeton, NJ, the Mobicedia box queries the FCC database for active tower locations in the area and stores the resulting figure in memory. Next, the script places objects indicating cellular towers across the figure. The script is as robust as it is functional; all of the original map tools provided with Google Maps – such as zoom, satellite, and hybrid view modes – work with the Mobicedia applet. The user can click on any of the objects representing towers, bringing up operator information, cellular licensing data, and contact phone numbers among other attributes. All in all, the PHP applet provided by Mobicedia works elegantly. The output of the above search is displayed on the next page.

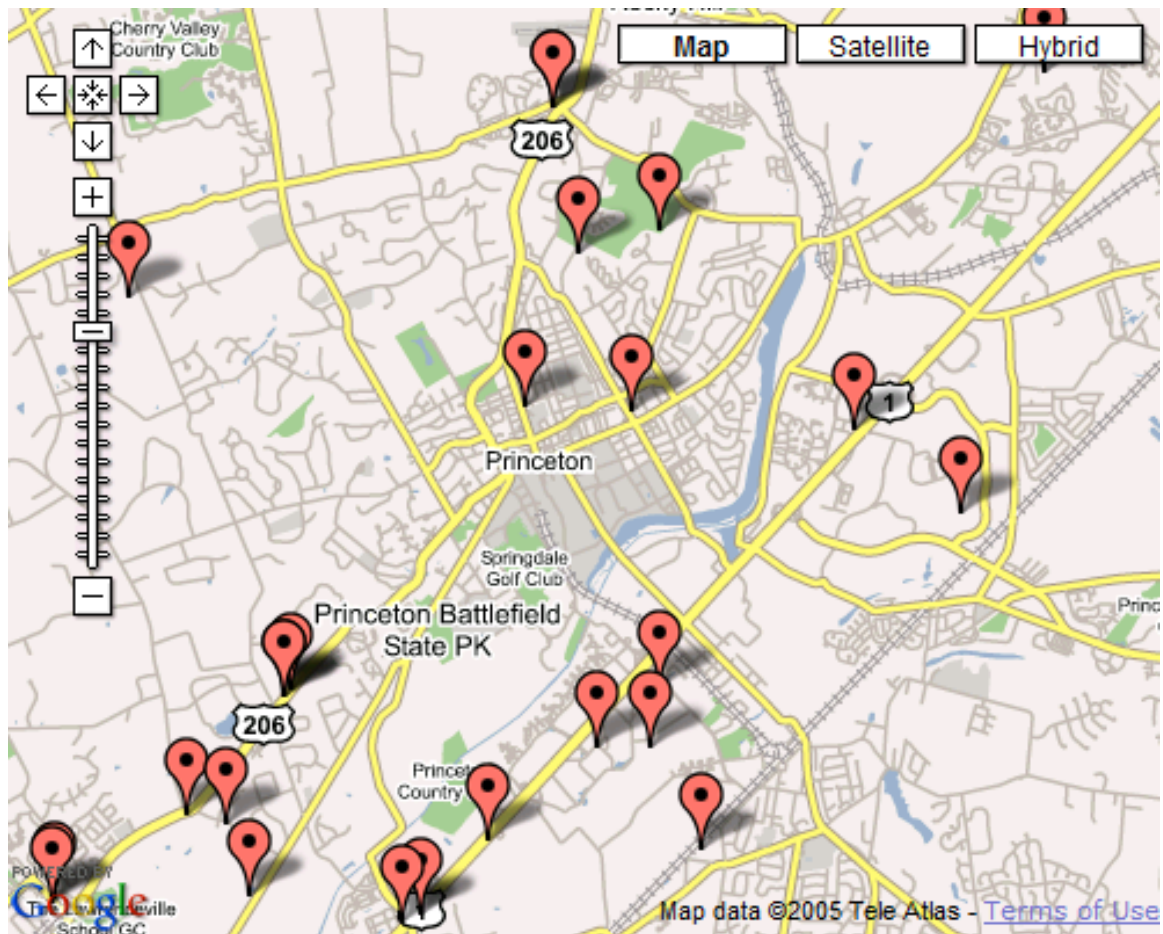


Figure 4.3. Output from Mobiledia search applet.

Notice that the shapes of the roads listed in the Mobiledia output are slightly different than the shapes of the roads in the Google Maps output itself. This occurs primarily because Mobiledia is referencing the TeleAtlas road database in its output while the previous Google Maps output references the NAVTEQ™ version of the database. The reasons for this difference are unknown.

The goal of SimulateSubscriber is to run steps (1) through (6) in some simulated environment that will reveal the usefulness of PathLocator, SoftSearch, and TowerLocator. However, the area directly around Princeton University is relatively crowded with a mess of towers and jumbled paths that would not do well in demonstrating the behavior of the procedures. Instead, consider the zoomed-in portion of US Route 1 slightly outside the Princeton campus as displayed on the next page:



Figure 4.4. BTSs and paths around US Route 1.

The map above could reasonably be approximated with the graph:

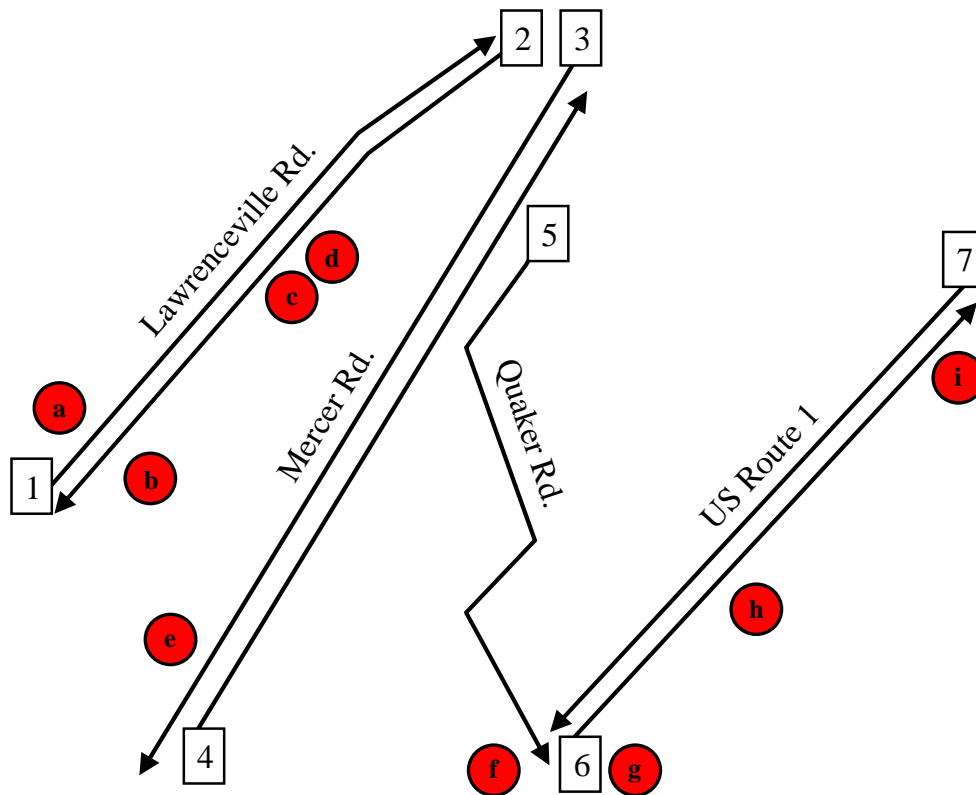


Figure 4.5. Diagram representation of Figure 4.4.

In Figure 4.5, each path is represented as an arrow labeled at its origin with a number from 1 to 7, while each BTS is represented as a red circle labeled with a letter from a to i . The directed arcs and BTS circles are an accurate representation of the map produced by Mobiledia; now the model needs the ability to draw it in some way.

SimulateSubscriber allows the user running the program to draw out paths and towers using Microsoft Excel’s interface and a virtual “drawing board”. The model divides a virtual square area into 100 rows and 100 columns to create this drawing board where the user can create custom paths and towers to her liking. Below is a portion of the virtual drawing board where the sample space Ω is created:

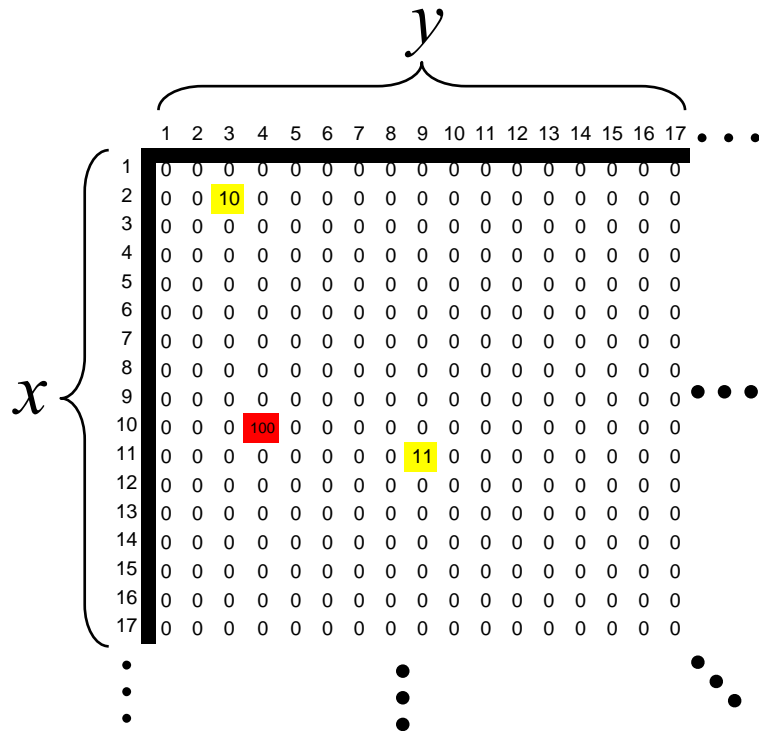


Figure 4.6. Virtual drawing board with 1 path and 1 tower.

The x coordinates on the drawing board are increasing with the rows while the y coordinates are increasing with the columns. For example, an entry in row 8, column 12 would have the coordinates $(8, 12)$. Notice then that this coordinate system is unlike the traditional Cartesian system where, in this case, x is increasing with the

columns while y is decreasing with the rows. This needs to be accounted for at some point during the calculation phase of the model when it iterates simulated travelers across all the possible paths in Ω .

While the paths in Figure 4.5 are denoted by single numbers ranging from 1 to 7, the path numbering system inherent in `SimulateSubscriber` and displayed on the drawing board in Figure 4.6 is slightly different and will be explained below. Consider Quaker Rd., listed as path 5 in Figure 4.5. The model numbers the path in the following way:

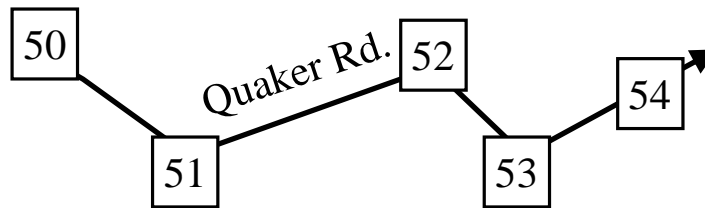


Figure 4.7. Path numbering method of `SimulateSubscriber`.

The path in Figure 4.7 has been rotated from Figure 4.5 for conciseness. Each numbered box above is referred to as a node in the model. Each pair of numbers – (50, 51) or (52, 53), for example – designates some minor arc of the path, where the sum of the minor arcs is the total path. The origin node for each path is always the path number multiplied by 10. For example, the path 3 origin node is 30; the path 7 origin node is 70, and so forth. Notice that in Figure 4.6 the entries $(2, 3) = 10$ and $(11, 9) = 11$; these entries designate that a path beginning at (2,3) and ending at (11,9) exists under Ω .

Towers in `SimulateSubscriber` are represented on the drawing board beginning with the index 100 and increasing in increments of 1. For example, in Figure 4.6, tower 1 exists at (10, 4) since it contains an entry of 100. The user can add additional towers at different points simply by entering in values of 101, 102, and so on according to her wishes. By using the drawing board feature of `SimulateSubscriber`, the user can successfully create a working representation of Figure 4.5 within the model. As a result,

the original traffic and handover patterns of the map in Figure 4.4 can be effectively simulated. Refer to the diagram below for an example of Figure 4.5 translated into the model using the drawing board feature of SimulateSubscriber:

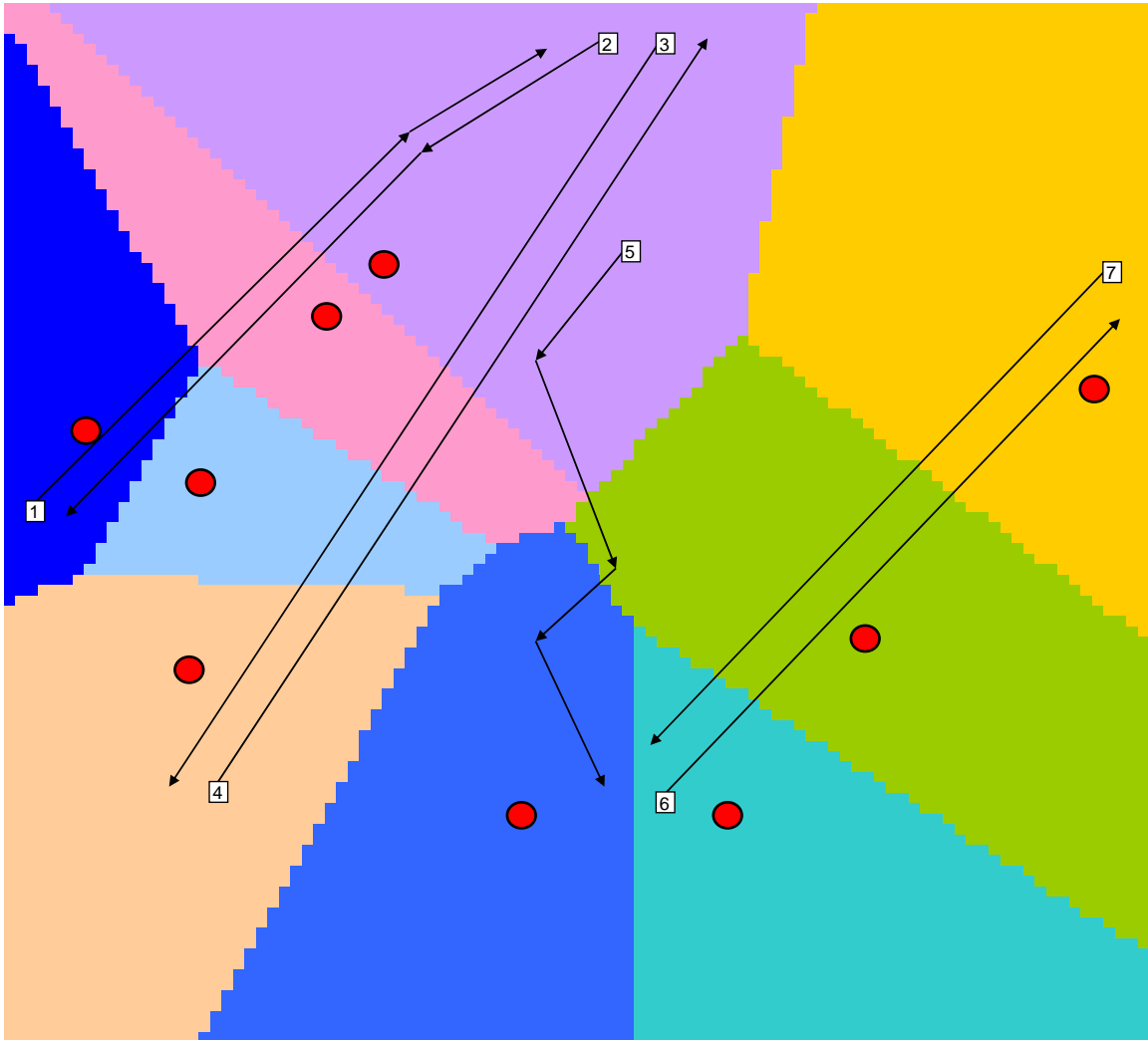


Figure 4.8. Translation of Figure 4.5 into the model.

Once the proper nodes are placed within the virtual drawing board, SimulateSubscriber makes several pass-through calculations to determine the dominate BTS at each point. Each separate color in Figure 4.8 represents a different BTS region, with handovers occurring when a user drives across a region border. The model begins with tower 100 and calculates the power P_r received at each point from BTS_{100} using the Friis free-space equation (Gibson 1183):

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \quad (4.1)$$

where,

$$\begin{aligned} P_t &= \text{the transmitted power} \\ G_t &= \text{the transmitted antenna gain} \\ G_r &= \text{the receiver antenna gain} \\ d &= \text{the distance between receiver and antenna} \\ L &= \text{the system losses, } L > 1 \\ \lambda &= \text{the wavelength of the signal in meters} \end{aligned} \quad (4.2)$$

The model then iterates sequentially across all other towers with the same formula to determine which tower provides the dominant signal at each point.

In reality, the Friis equation is clumsy and impractical to implement for the purposes of this paper, so Gibson suggests an alternate equation to approximate the path loss $\overline{PL}(d)$ of a BTS signal as a function of distance d between transmitter and receiver (1188):

$$\overline{PL}(d) \propto \left(\frac{d}{d_0} \right)^n \quad (4.3)$$

where n is some path loss exponent. Gibson recommends a path loss exponent of 4 to simulate real world conditions. `SimulateSubscriber` will, for the purposes of this paper, use the inverse path loss equation to calculate the signal strength $P(i, d)$ of BTS_i at a distance d away:

$$P(i, d) = \frac{P_0(i)}{d^4} \quad (4.4)$$

where $P_0(i)$ is the broadcast power of BTS_i and

$$P_0(i) = N \sim (\mu, \sigma^2) \quad (4.5)$$

for some calibrated μ and σ . It should be noted that the model is robust enough to handle any path loss exponent, though an exponent of 4 will be used for all simulations done here.

Now that `SimulateSubscriber` has built its state space Ω , it creates a virtual traveler $T_{s,p,a,b}$ where,

$$\begin{aligned}
 T_s &\sim N(\mu_s, \sigma_s^2) = \text{the driving speed of } T_{s,p,a,b} \\
 T_p &= \{p_0 : p_0 \sim \text{Unif}(1, |\Theta(T)|), p_0 \in \mathbb{Z}\} = \text{the starting path of } T_{s,p,a,b} \\
 T_a &= \begin{cases} 1, & \text{if } T_{s,p,a,b} \text{ is active} \\ 2, & \text{if } T_{s,p,a,b} \text{ is passive} \end{cases} \\
 t_b &\sim N(\mu_b, \sigma_b^2) = \text{the beacon interval (time) of } T_{s,p,a,b}
 \end{aligned} \tag{4.6}$$

Note that a value of $a = 1$ indicates that the subscriber is actively talking on her phone, meaning that BTS handovers will occur at the instant she crosses over into another BTS region, as that BTS will provide stronger signal at that instant. By contrast, a value of $a = 2$ means that the subscriber is not talking on her phone. The user must wait a specified amount of time — called the beacon interval t_b — until her handset sends a measurement report out to be considered for handover. The difference between active and passive handovers is diagrammed below:

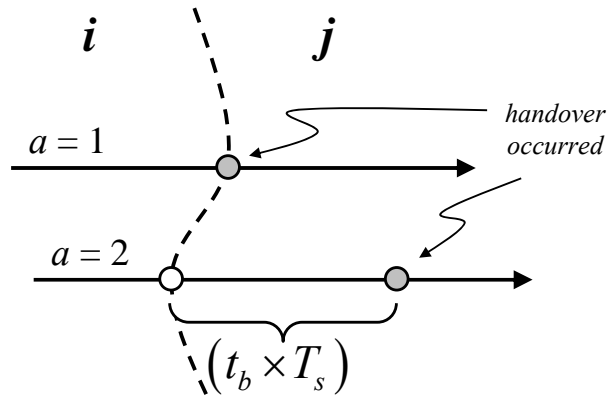


Figure 4.9. Active vs. passive travelers.

Figure 4.9 illustrates the different handover behavior between active and passive travelers under Ω . While the active traveler ($a = 1$) is handed over from BTS_i to BTS_j at the border of the two cells, the passive traveler ($a = 2$) must wait until a measurement report is sent to begin receiving signal from BTS_j . In effect, the handover location of the passive user is actually the error term ($t_b \times T_s$) away from the cell border. This error term complicates the desired path and trip matching as will be demonstrated later.

Now that `SimulateSubscriber` has created a virtual traveler $T_{s,p,a,b}$ on some starting node, it begins to move the traveler in increments with respect to time dt . Because the coordinate system in the model is slightly different than the traditional Cartesian system used in mathematics, the traveler's direction along each path must be calculated somewhat tediously. Remember that the x coordinate of $T_{s,p,a,b}$ is increasing with the rows while the y coordinate is increasing with the columns. Consider a traveler driving from node i to node j . Let

$$s(T_{s,p,a,b}, t) = (x, y) \quad (4.7)$$

be the position of $T_{s,p,a,b}$ at time t and

$$\begin{aligned} \lambda_{x,i} &= \text{the } x \text{ coordinate of node } i \\ \lambda_{y,i} &= \text{the } y \text{ coordinate of node } i \\ \lambda_{x,j} &= \text{the } x \text{ coordinate of node } j \\ \lambda_{y,j} &= \text{the } y \text{ coordinate of node } j \end{aligned} \quad (4.8)$$

then `SimulateSubscriber` creates the direction variables

$$\delta_x = \begin{cases} \text{sgn}(\lambda_{y,j} - \lambda_{y,i}), & \text{if } \lambda_{x,i} = \lambda_{x,j} \\ \text{sgn}(\lambda_{x,j} - \lambda_{x,i}) \left| \left(\sin \left(-\arctan \left(\frac{\lambda_{x,i} - \lambda_{x,j}}{\lambda_{y,j} - \lambda_{y,i}} \right) \right) \right) \right|, & \text{if } \lambda_{x,i} \neq \lambda_{x,j} \end{cases} \quad (4.9)$$

and

$$\delta_y = \begin{cases} \operatorname{sgn}(\lambda_{x,j} - \lambda_{x,i}), & \text{if } \lambda_{y,i} = \lambda_{y,j} \\ \operatorname{sgn}(\lambda_{y,j} - \lambda_{y,i}) \left| \cos \left(-\arctan \left(\frac{\lambda_{x,i} - \lambda_{x,j}}{\lambda_{y,j} - \lambda_{y,i}} \right) \right) \right|, & \text{if } \lambda_{x,i} \neq \lambda_{x,j} \end{cases} \quad (4.10)$$

Next, the model updates the position of the virtual traveler $T_{s,p,a,b}$ iteratively with

$$s(T_{s,p,a,b}, t + dt) = \left(s_x(T_{s,p,a,b}, t) + (\delta_x T_s dt), s_y(T_{s,p,a,b}, t) + (\delta_y T_s dt) \right) \quad (4.11)$$

and records the BTS handovers if they occur at any time interval dt . For the state space Ω defined in Figure 4.5, a portion of the SimulateSubscriber output is listed below:

| Car Info | | | | |
|-----------|-------|------|------------|-----------------|
| Car Index | Speed | Path | Total Time | Active/Passive? |
| 1 | 0.55 | 1 | 112.4 | Active |
| 2 | 0.67 | 6 | 89.47 | Active |
| 3 | 0.58 | 6 | 102.74 | Active |
| 4 | 0.77 | 4 | 108.83 | Passive |
| 5 | 0.64 | 2 | 100.94 | Passive |

Figure 4.10. Car Info section of SimulateSubscriber output.

| Tower Switching (Handover) | | | | |
|----------------------------|------------|-----------------------|------------------------|------------|
| Start Tower | Next Tower | Actual Crossover Time | Passive Crossover Time | Difference |
| 1 | 3 | 38.58 | | |
| 7 | 8 | 17.06 | | |
| 7 | 8 | 21.58 | | |
| 5 | 2 | 29.58 | 30.90 | 1.32 |
| 4 | 3 | 49.76 | 50.43 | 0.67 |

Figure 4.11. Handover time section of SimulateSubscriber output.

| Path Guessing | | | | | | |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| Path 1 Possible? | Path 2 Possible? | Path 3 Possible? | Path 4 Possible? | Path 5 Possible? | Path 6 Possible? | Path 7 Possible? |
| YES | NO | NO | NO | NO | NO | NO |
| NO | NO | NO | NO | NO | YES | NO |
| NO | NO | NO | NO | NO | YES | NO |
| NO | NO | NO | YES | NO | NO | NO |
| NO | YES | NO | NO | NO | NO | NO |

Figure 4.12. PathLocator and SoftSearch section of SimulateSubscriber output.

| Handover Location | | | |
|-------------------|------------|------------|------------|
| Start Tower | Next Tower | X Position | Y Position |
| 1 | 3 | 27.34 | 22.32 |
| 7 | 8 | 25.10 | 28.00 |
| 7 | 8 | 25.12 | 28.10 |
| 5 | 2 | 45.48 | 80.00 |
| 4 | 3 | 25.10 | 28.00 |

Figure 4.13. Handover time section of `SimulateSubscriber` output.

The above outputs are a result of `SimulateSubscriber` running with each BTS broadcasting at a level P_0 with

$$P_0(i) = N \sim (\mu, 0) \quad (4.12)$$

Incidentally, this is the method by which $\varphi(P)$ and C_{adjacent} are initialized within the model. That is, σ is set to 0 for all calculations of $P_0(i)$, and the resulting cell-border structure is used to record handovers for each path θ_i and determine which paths are feasible using $\varphi(P)$ and C_{adjacent} . Refer to the end of Appendix A for pictures displaying the effect of σ on the model's graphical output.

Since the variables dedicated to `PathLocator` and `SoftSearch` are initialized with $\sigma = 0$ in P_0 , it comes as no surprise that the path-guessing in Figure 4.12 is perfect since the handover behavior in Figure 4.10 and Figure 4.11 is directly related to P_0 . In effect, `SimulateSubscriber` is unable to test the effectiveness of `PathLocator` versus `Softsearch` when $\sigma = 0$ since they both guess perfectly every time. By increasing σ and simulating a large ($N = 1000$) number of drivers in Ω , the weakness of the `PathLocator` algorithm versus `SoftSearch` begins to display itself:

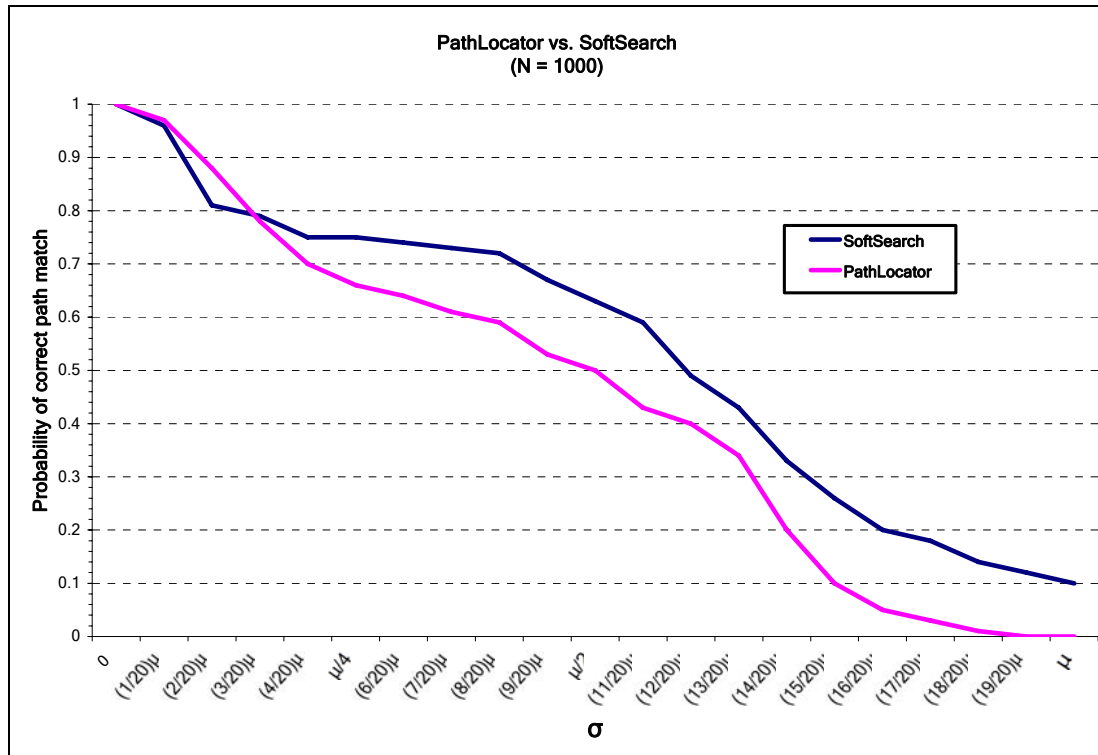


Figure 4.14. PathLocator vs. Softsearch performance on Ω from Figure 4.5.

While BTSs would never realistically transmit signal with the σ displayed in the graph above, SoftSearch does perform slightly better than PathLocator under these extreme circumstances. Consider instead the following, more realistic state space Ω :

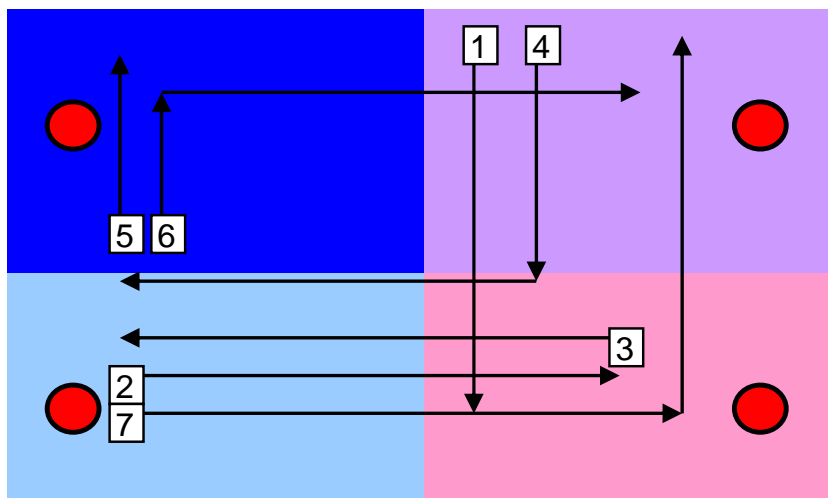


Figure 4.15. Example state space Ω representing an urban area.

Figure 4.15 composes a typical intersection in any major metropolitan area; drivers can turn left, right, or stay straight at the (implied) streetlight at the center. This is where the true power of SoftSearch can be found, in the heart of realistic application:

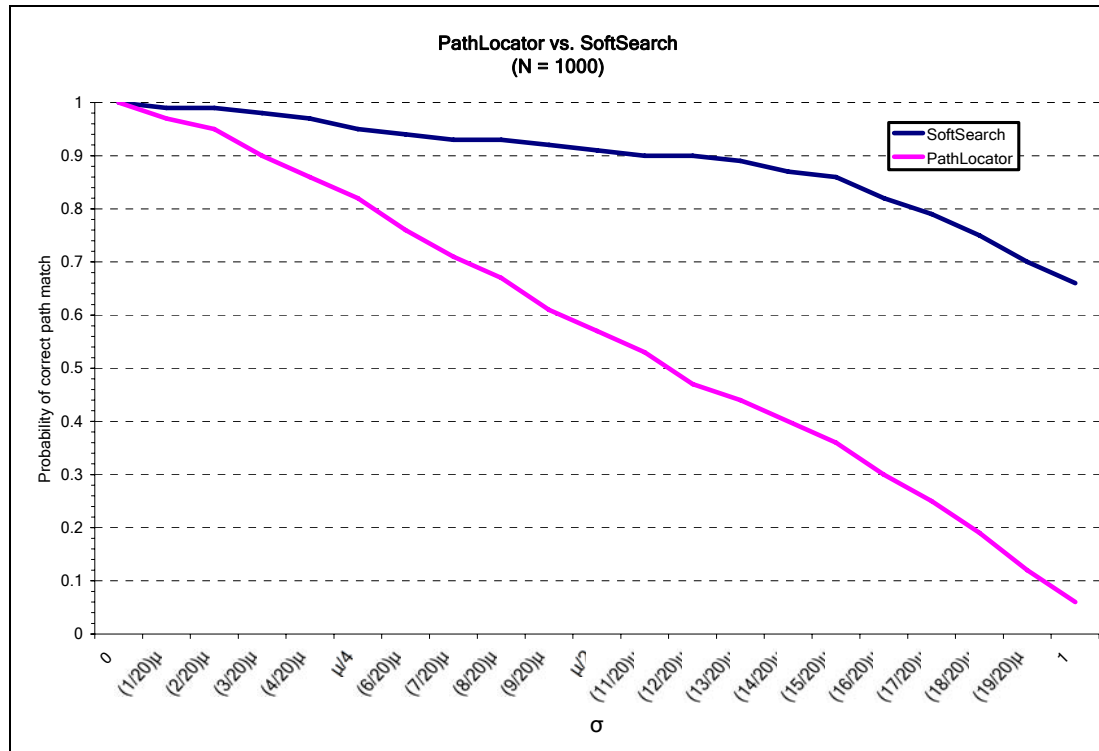


Figure 4.16. PathLocator vs. Softsearch performance on Ω from Figure 4.15.

Needless to say, SoftSearch outperforms PathLocator in every situation.

A Glance at TowerLocator

The mathematics behind TowerLocator are elementary at best, but their implementation and demonstration here will be included for the sake of completeness. Consider again the state space Ω described in Figure 3.12. Then their representation in SimulateSubscriber and relevant TowerLocator output is:

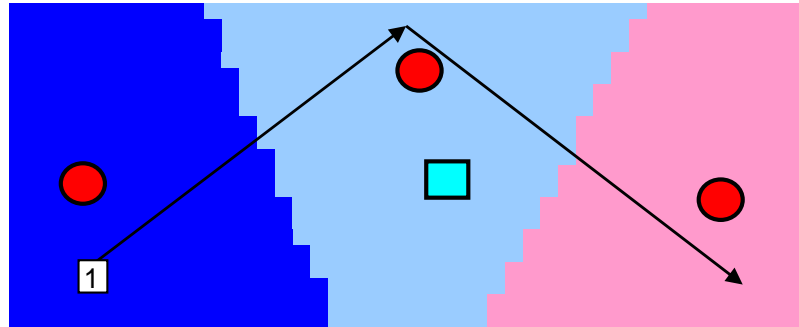


Figure 4.17. SimulateSubscriber representation of Figure 3.12.

Note that the turquoise square at the center of the diagram is successfully approximating the center BTS position as shown in Figure 3.12. The raw data associated with the analysis is listed below.

| Car Index | 1st Handover | | 2nd Handover | |
|-----------|--------------|------------|--------------|------------|
| | X Position | Y Position | X Position | Y Position |
| 1 | 19.58 | 41.00 | 20.50 | 59.00 |
| 2 | 18.93 | 41.79 | 20.97 | 59.56 |
| 3 | 19.58 | 41.00 | 20.50 | 59.00 |
| 4 | 19.32 | 41.31 | 20.84 | 59.40 |
| 5 | 19.48 | 41.13 | 20.67 | 59.21 |

Figure 4.18. TowerLocator data.

| Least Squares Point | | | | | Actual Tower Location | |
|---------------------|------------|----------|----------|-------------|-----------------------|------------|
| X Position | Y Position | X Error | Y Error | Total Error | X Position | Y Position |
| 20 | 50.2 | 4.828635 | 809.5728 | 814.401414 | 16 | 50 |

Figure 4.19. TowerLocator results.

“Aging isn’t that bad if you consider the alternatives.”

--Maurice Chevalier

5 Alternatives and the Financial Case

Information, in today’s business world, is worth more than its weight in gold if it is correct, reliable, and consistent. While dominant companies are currently capitalizing on algorithms developed to deliver information to the consumers who desire it most, enterprising minds are constantly searching for the next big thing in information engineering. A search of the words, “cellular” and “location” in the United States Patent and Trademark Office online registration database returns 59,618 patents on the subject. The alternatives to the next generation of cellular subscriber location range from promising, to interesting, to ludicrous.

For a refreshing instance of the latter, take, for example, US Patent #5,572,221 by Marlevi, et al. As a replacement to current GPS and possible future handover-based path estimation systems, Marlevi outlines a:

method of predicting a next location of a mobile terminal based on stored previous locations of the mobile terminal includes the step of comparing a current sequence that includes the current location of the mobile terminal and a plurality of previous locations of the mobile terminal to each of a plurality of stored sequences that each include previous locations of the mobile terminal. (USPTO)

So, Marlevi's brilliant idea is to create for a cellular subscriber a database containing all of her past destinations that would hopefully hold enough information to predict where she would be next. The database, as described later in the patent, is opt-in, however, meaning that a given user's path could only be estimated if they were to enter their path in the database in the first place! Surely this cannot be a viable business opportunity.

On the other hand, some registered patents bring about very exciting ideas and methods that could actually help create new revenue streams for existing telecoms. US Patent # 6,249,680 by Wax, Mati, Hilsenrath, Oliver, Bar, and Abraham outlines a viable method of using existing multipath to invent

...an array of p antennas belonging to a cellular network base station. A location finding apparatus connected to the base station contains a multichannel receiver that uses PN sequence information provided by the base station receiver to despread the p signals and to separate each of the p signals into temporally distinct multipath parts. A signal processor calculates a signal signature for each active mobile. The signature is comprised of a code correlation function, a set of temporal delays corresponding to the multipath parts and a set of signal subspaces. The signature is then compared to a database of calibrated signal signatures and corresponding locations, and a location whose calibrated signature best matches the measured signature is selected as the most likely location of the mobile transmitter. (USPTO)

The method outlined seems feasible since multipath is an inherent characteristic of all modern cellular networks, and the idea seems exciting if it would actually enable ubiquitous and effective user location. Hopefully, the cost of the antenna array and associated equipment do not keep this technology from reaching the mainstream.

But Is it Realistic?

While it is true that a cellular replacement for GPS would mean big business and big convenience, it is perhaps too early in today's current technological state to expect anything of the sort in the near future. Consider the handover-based method discussed

in chapter 3 and modeled in chapter 4 of this paper. In reality, the performance of the technology – even under ideal circumstances – left a large amount to be desired. When modeling user travel and the associated handover around US Route 1 in Princeton, for example, the results of PathLocator and Softsearch only provided correct path matches in around 40%-60% of the cases. This low threshold of success is much too unacceptable for any sort of real-world implementation. In addition, the fact that telecoms are deathly afraid of parting with their valuable user data strikes yet another mortal blow for the birth of any sort of handover-based technology. Perhaps, considering the many obstacles currently preventing cellular path estimation and its sister technologies, GPS really will be the dominant standard for decades to come.

But that is no reason to stop trying. The technological landscape has always been shaped by those few rogues that refuse to accept the status quo as the perpetual nature of things. The ludicrous idea of today could just as easily become the breakthrough revelation of tomorrow, so we must constantly push to test the boundaries of social and technological norms if we are to hope for any progress. Take the small company NET discussed in this paper. Even if their Cell Probe technology could never be realistically implemented, it is their kind of thought that brings about a brighter and more technically advanced future. It is the vision of small companies like these that fuel the standards of tomorrow.

Today, millions of people sit in traffic every day due to a lack of information. The current methods of user location under cellular can cost as much as \$0.01 per use, yet their implementation would allow for perfect detection of traffic patterns, saving millions of wasted man-hours and increasing productivity worldwide (Kornhauser). If someone were to push a working cellular location technology through to the dawdling telecom industry, perhaps they could capitalize on this. If one were to invent path-estimation technology that would only cost the user or provider one-thousandth or even one-

millionth of a cent to use, that person would effectively eliminate any unneeded traffic as long as cellular systems existed. That person would be a hero in the eyes of the morning commuter, the knight that defeated the dark animal of gridlock. That person would also become very rich, and there's nothing bad about that.

"I just wrote a book, but don't go out and buy it yet, because I don't think it's finished yet."
--Lawrence Welk

Bibliography and References

"About Us." Movabletype. 14 Apr. 2006 <<http://www.movabletype.com/about/index.html>>.

Bartlett, John. Bartlett's Familiar Quotations, Fifteenth Edition. Boston, MA: Little, Brown and Company, 1980.

Bernatchez, Eric. "What Difference Does It Make If I Use a CDMA, GSM, or TDMA Phone?" About. 14 Apr. 2006
<http://cellphones.about.com/od/thecellularfaq/f/cf_technologies.htm>.

"Cell Phone Facts and Statistics." NetworkWorld. 07 Feb. 2001. 9 Apr. 2006
<<http://www.networkworld.com/research/2001/0702featside.html>>.

Cooley, J. D. Wireless FAQ (Revision A.5). alt.cellular Newsgroups. 2005. 11 Apr. 2006
<http://www.wirelessadvisor.com/wireless_faq.cfm>.

"FCC Antenna Structure Registration (ASR)." Federal Communications Commission. 14 Apr. 2006 <<http://wireless.fcc.gov/antenna/>>.

Filipiak, Janusz. Real Time Network Management. New York, NY: Elsevier Science Company, Inc., 1991.

Gibson, Jerry D. The Communications Handbook. Salem, MA: CRC P, 1997.

"Google Local." Google. 14 Apr. 2006 <<http://maps.google.com/>>.

Hills, M. T., and B. G. Evans. Telecommunication Systems Design Volume 1: Transmission Systems. London: George Allen & Unwin LTD, 1979.

Horak, Ray, and Mark A. Miller. Communications Systems & Networks. New York, NY: M&T Books, 1997.

"IEC: Global System for Mobile Communication (GSM)." International Engineering Consortium. 10 Apr. 2006
<<http://www.iec.org/online/tutorials/gsm/topic03.html?Next.x=31&Next.y=14&Next=Next>>.

"IEC: Time Division Multiple Access (TDMA)." International Engineering Consortium. 9 Apr. 2006 <<http://www.iec.org/online/tutorials/tdma/>>.

Kornhauser, Alain L. Personal interview. 9 Mar. 2006.

Marlevi. "United States Patent: 5,572,221." USPTO Patent Full-Text and Image Database. 5 Nov. 1996. USPTO. 16 Apr. 2006

"Patent Full-Text and Full-Image Databases." United States Patent and Trademark Office. USPTO. 16 Apr. 2006 <<http://www.uspto.gov/patft/index.html>>.

Paulraj, Arogyaswami, Vwani Roychowdhury, and Charles D. Schaper. Communications, Computation, Control and Signal Processing. Norwell: Kluwer Academic, 1997.

Prasad, Ramjee, Werner Mohr, and Walter Konhauser. Third Generation Mobile Communication Systems. Norwood, MA: Artech House, 2000.

"Rayleigh Fading." Wikipedia. 11 Apr. 2006
<http://en.wikipedia.org/wiki/Rayleigh_fading>.

Redl, Siegmund M., Matthias K. Weber, and Malcolm W. Oliphant. An Introduction to GSM. Norwood, MA: Artech House, 1995.

Saadawi, Tarek N., Mostafa H. Ammar, and Ahmed El Hakeem. Fundamentals of Telecommunication Networks. New York, NY: John Wiley & Sons, Inc., 1994.

Schwartz, Mischa, William R. Bennett, and Seymour Stein. Communication Systems and Techniques. New York, NY: IEEE Press, 1996.

"TomTom Press Release." 26 2005. TomTom NV. 11 Apr 2006
<<http://www.tomtom.com/investor/press.php?ID=147&Year=2005&Language=4&TT=f69249f40efd2ecd170acd902fe144e1>>.

Wax, Mati, Hilsenrath, Oliver, Bar, and Abraham. "United States Patent: 6,249,680." USPTO Patent Full-Text and Image Database. 19 June 2001. USPTO. 16 Apr. 2006.

"Yahoo! Finance." 09 Apr. 2006. Yahoo! 09 Apr. 2006
<<http://finance.yahoo.com/q/bc?s=GOOG&t=2y&l=on&z=m&q=l&c=>>>.

“Any sufficiently advanced bug is indistinguishable from a feature.”
--Bruce Brown

Appendix A

Procedure SimulateSubscriber

```
Sub SimulateSubscriber()
Randomize
'Global variables
Dim currX, currY As Integer
Dim Paths(100, 100) As Integer 'Remember that each path can only have 9 nodes (i.e. 10 to 18)
Dim DifferentPaths, i, j, k As Integer
Dim NodesX(), NodesY() As Integer 'VB cannot handle objects
Dim Seed As Single
Dim Time, ElapsedTime, dt As Double

'NodesX and NodesY work as follows
'A value of 53 in row 7, column 8 is:
'NodesX(5, 4) = 7
'NodesY(5, 4) = 8
'This describes the 4th entry in the 5th path (since 50 is an entry)

'Car variables
Dim XPos, YPos, Speed, XDirection, YDirection As Double
Dim SpeedMean, SpeedSD As Double
Dim Precision As Single
Dim CurrentCar, NumCars, CurrentNode, NextNode, FinalNode As Integer
Dim Active, Passive, CurrentState As Integer 'Is passenger talking on phone?
Dim CurrentTower As Integer
Dim OldTowers As Integer 'How many towers have been visited already?
Dim BeaconInterval As Double 'For Passive drivers
Dim BeaconIntervalMean, BeaconIntervalSD As Double
'NumCars is the number of cars to simulate
'CurNode is the last visited node of the current car

'Tower variables
'Must have at least one tower for macro to function correctly!
Dim TowersX(), TowersY() As Integer
Dim TowerStrength(100, 100) As Double
Dim TowerCoverage(100, 100) As Integer 'Which tower is currently servicing the car?
Dim DifferentTowers As Integer
```

```

Dim TowerOutputMean, TowerOutputSD, TowerOutput As Double      'How much power each tower outputs
                                                                (all the same amount)
Dim CurrentDistance As Double
Dim OrderOfDecay As Double      'Rate at which tower power output decays (1/distance^order)
Dim TowerSetupOnly As Integer
Dim PaintArea As Integer      'Determine whether to repaint sheets after every iteration

'Path-guessing variables
Dim PathsConsidered() As Integer
Dim FeasibleSet(9) As Integer      'Possible Paths
Dim GuessPaths As Integer      'Determine whether or not to guess paths after car iteration
Dim CurrentEvaluation As Integer
Dim SoftEstimation As Integer      'Soft estimation includes any path containing actual path handovers
Dim SoftPath() As Integer      'Use in soft estimation
Dim SoftPathLength, CurrentPathLength As Integer
Dim SoftPathTries As Integer      'SoftPath fits into PathsConsidered SoftPathTries times
Dim Offset As Integer
Dim Found As Boolean

'For simplicity's sake, path-guessing algorithm has access to variables it could
'otherwise calculate (i.e. DifferentPaths)

'Dialogue boxes for user input
TowerSetupOnly = MsgBox("Tower setup only (no iteration of cars)?", vbYesNo)

If (TowerSetupOnly = vbNo) Then

    GuessPaths = MsgBox("Guess paths after car iteration?", vbYesNo)

    If (GuessPaths = vbYes) Then

        SoftEstimation = MsgBox("Use soft estimation for path guessing?", vbYesNo)

    End If

    PaintArea = MsgBox("Repaint area after every iteration?", vbYesNo)

End If

If (TowerSetupOnly = vbNo) Then

    NumCars = InputBox("Number of cars to iterate:", "Set Variable: NumCars", 10)
    SpeedMean = InputBox("Mean car speed:", "Set Variable: SpeedMean", 0.5)
    SpeedSD = InputBox("Mean car speed standard deviation:", "Set Variable: SpeedSD", 0.1)
    TowerOutput = InputBox("Tower output:", "Set Variable: TowerOutput", 10000)
    OrderOfDecay = InputBox("Order of decay for tower output (1/distance^order):", "Set Variable:
                                                                    OrderOfDecay", 6)

    dt = InputBox("Value for dt:", "Set Variable: dt", 0.01)
    Precision = InputBox("Precision (when (Error < Precision), assume car is at (Node+1):", "Set
                                                                    Variable: Precision", 0.1)

End If

If (TowerSetupOnly = vbYes) Then

    'Initialize global variables so macro works quickly
    Time = 0
    ElapsedTime = 0 'Use to calculate handover time for Passive drivers
    dt = 0.01
    Precision = 0.1

    'Initialize car variables
    'Speed is N ~ (SpeedMean, SpeedSD^2)
    SpeedMean = 0.5
    SpeedSD = 0.1
    NumCars = 2
    Active = 2      'Arbitrary value, driver is talking on phone
    Passive = 1    'Arbitrary value, driver is not talking on phone
    BeaconIntervalMean = 2 'Values for Passive drivers only, in units of TIME
    BeaconIntervalSD = 0.2 'Use a relatively high SD since cell phones vary

    'Initialize tower variables
    TowerOutputMean = 100000
    TowerOutputSD = 10000
    OrderOfDecay = 4
    OldTowers = 0

```

```

End If

'Clear existing paths
Worksheets("Tower Coverage").Select
ActiveSheet.Shapes.SelectAll
Selection.Delete
Worksheets("Paths").Select
ActiveSheet.Shapes.SelectAll
Selection.Delete

'Clear existing Tower Results data
Worksheets("Tower Results").Select
Range("A3:AA50000").Select
Selection.Font.ColorIndex = 1 'Reset white values
Selection.Clear
Range("A3").Select

'Clear existing Car Results data
Worksheets("Car Results").Select
Range("A3:CC50000").Select
Selection.Font.ColorIndex = 1 'Reset white values
Selection.Clear
Range("A3").Select

'Read all values into array
Worksheets("Paths").Select
Range("B2").Select
For currX = 1 To 100

    For currY = 1 To 100

        Paths(currX, currY) = ActiveCell.Offset(currX - 1, currY - 1)

    Next currY

Next currX

'Now determine how many different paths exist (up to 9)
DifferentPaths = 0
For currX = 1 To 100

    For currY = 1 To 100

        If ((Paths(currX, currY) > DifferentPaths) And (Paths(currX, currY) < 100)) Then

            DifferentPaths = Paths(currX, currY)

        End If

    Next currY

Next currX

DifferentPaths = Int(DifferentPaths / 10)

'Determine how many different towers exist (up to 99)
DifferentTowers = 0
For currX = 1 To 100

    For currY = 1 To 100

        If ((Paths(currX, currY) > DifferentTowers) And (Paths(currX, currY) >= 100)) Then

            DifferentTowers = Paths(currX, currY)

        End If

    Next currY

Next currX
DifferentTowers = (DifferentTowers - 99)

'Create the Nodes arrays and initialize them with all 0 values
ReDim NodesX(DifferentPaths, 10)
ReDim NodesY(DifferentPaths, 10)
ReDim TowersX(DifferentTowers) 'Tower arrays are one dimensional
ReDim TowersY(DifferentTowers) 'because individual towers are independent

For currX = 1 To DifferentPaths

```

```

For currY = 1 To 10
    NodesX(currX, currY) = 0
    NodesY(currX, currY) = 0
Next currY
Next currX

For i = 1 To DifferentTowers
    TowersX(i) = 0
    TowersY(i) = 0
Next i

'Populate the Nodes arrays with the appropriate values
For currX = 1 To 100
    For currY = 1 To 100
        If ((Paths(currX, currY) > 0) And (Paths(currX, currY) < 100)) Then 'Square is a path
            NodesX(Int(Paths(currX, currY) / 10), (Paths(currX, currY) - ((Int(Paths(currX, currY) / 10)) * 10)) + 1) = currX
            NodesY(Int(Paths(currX, currY) / 10), (Paths(currX, currY) - ((Int(Paths(currX, currY) / 10)) * 10)) + 1) = currY
        End If
        If Paths(currX, currY) >= 100 Then 'Square is a tower
            TowersX(Paths(currX, currY) - 99) = currX
            TowersY(Paths(currX, currY) - 99) = currY
        End If
    Next currY
Next currX

'Draw path arrows on appropriate graphs
For i = 1 To DifferentPaths
    FinalNode = 0
    For CurrentNode = 1 To 10
        If NodesX(i, CurrentNode) = 0 Then
            FinalNode = CurrentNode - 1
            Exit For
        End If
    Next CurrentNode
    For CurrentNode = 1 To (FinalNode - 1)
        Worksheets("Paths").Select
        With ActiveSheet.Shapes.AddLine((((NodesY(i, CurrentNode) + 1) * 6) - 3), (((NodesX(i, CurrentNode) + 1) * 6) - 3), (((NodesY(i, CurrentNode + 1) + 1) * 6) - 3), (((NodesX(i, CurrentNode + 1) + 1) * 6) - 3)).Line
            .DashStyle = msoLineSolid
            .ForeColor.RGB = RGB(0, 0, 0)
            .Weight = 1
            .EndArrowheadStyle = msoArrowheadTriangle
            .EndArrowheadLength = msoArrowheadLengthMedium
            .EndArrowheadWidth = msoArrowheadWidthMedium
        End With
        Worksheets("Tower Coverage").Select
        With ActiveSheet.Shapes.AddLine((((NodesY(i, CurrentNode) + 1) * 6) - 3), (((NodesX(i, CurrentNode) + 1) * 6) - 3), (((NodesY(i, CurrentNode + 1) + 1) * 6) - 3), (((NodesX(i, CurrentNode + 1) + 1) * 6) - 3)).Line
            .DashStyle = msoLineSolid
            .ForeColor.RGB = RGB(0, 0, 0)
            .Weight = 1
            .EndArrowheadStyle = msoArrowheadTriangle
            .EndArrowheadLength = msoArrowheadLengthMedium
            .EndArrowheadWidth = msoArrowheadWidthMedium
        End With
    Next CurrentNode
End For

```

```

Next CurrentNode

'Label path with a textbox
ActiveSheet.Shapes.AddTextbox(msoTextOrientationHorizontal, NodesY(i, 1) * 6, NodesX(i, 1) * 6,
                             10, 12).Select

Selection.Characters.Text = "" & i
With Selection.Characters(Start:=1, Length:=1).Font
    .Name = "Arial"
    .FontStyle = "Regular"
    .Size = 10
    .Strikethrough = False
    .Superscript = False
    .Subscript = False
    .OutlineFont = False
    .Shadow = False
    .Underline = xlUnderlineStyleNone
    .ColorIndex = xlAutomatic
End With
Next i

'Draw towers on "Tower Coverage"
Worksheets("Tower Coverage").Select
For i = 1 To DifferentTowers

    ActiveSheet.Shapes.AddShape(msoShapeOval, (((TowersY(i) + 1) * 6) - 12), (((TowersX(i) + 1) *
                                                                                   6) - 12), 15, 15).Select

    With Selection.ShapeRange
        .Fill.Visible = msoTrue
        .Fill.Solid
        .Fill.ForeColor.SchemeColor = 10
        .Fill.Transparency = 0#
        .Line.Weight = 1.5
        .Line.DashStyle = msoLineSolid
        .Line.Style = msoLineSingle
        .Line.Transparency = 0#
        .Line.Visible = msoTrue
        .Line.ForeColor.SchemeColor = 64
        .Line.BackColor.RGB = RGB(255, 255, 255)
    End With

Next i

'Clean up area
Worksheets("Tower Coverage").Select
Range("B2:CW101").Select
With Selection
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlCenter
    .WrapText = False
    .Orientation = 0
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
End With
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .Weight = xlThick
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThick
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThick
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThick
    .ColorIndex = xlAutomatic
End With
Selection.Borders(xlInsideVertical).LineStyle = xlNone

```

```

Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
Range("B2").Select

Worksheets("Paths").Select
Range("B2:CW101").Select
With Selection
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlCenter
    .WrapText = False
    .Orientation = 0
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
End With
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .Weight = xlThick
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThick
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThick
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThick
    .ColorIndex = xlAutomatic
End With
Selection.Borders(xlInsideVertical).LineStyle = xlNone
Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
Range("B2").Select

'Now that we know where the towers are, populate the "Tower Strength" and "Tower Coverage" sheets
'We need to do this iteratively since TowerOutput is now random

For CurrentCar = 1 To NumCars 'Iterate over all cars
For i = 1 To DifferentTowers
    'Calculate tower strength
    Sheets("Current Parameters").Range("C7").Formula = "=NORMINV(" & Rnd() & "," & TowerOutputMean &
        "," & TowerOutputSD & ")" 'Inverse Normal function
    TowerOutput = Sheets("Current Parameters").Range("C7").Value

    For currX = 1 To 100
        For currY = 1 To 100
            CurrentDistance = (((TowersX(i) - currX) ^ 2) + ((TowersY(i) - currY) ^ 2)) ^ (1 / 2) + 1
                'Assume that the car is not actually on a tower

            If (i = 1) Then 'Initialize all values to first tower then compare when i > 1
                If (CurrentDistance > 0) Then
                    Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value = TowerOutput /
                        (CurrentDistance ^ OrderOfDecay)

                End If

                If (CurrentDistance = 0) Then
                    Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value = TowerOutput

                End If

                Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = 1

            End If
        End For
    End For
End For

```

```

    If (i > 1) Then
        If (CurrentDistance > 0) Then 'Only replace if current tower is more powerful than
            'existing tower
            If ((TowerOutput / (CurrentDistance ^ OrderOfDecay)) > Worksheets("Tower
            Strength").Cells(currX + 1, currY + 1).Value) Then

                Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value = TowerOutput /
                (CurrentDistance ^ OrderOfDecay)

                Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = i

            End If
        End If

        If (CurrentDistance = 0) Then 'Obviously no other tower is stronger at these
            'coordinates

            Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value = TowerOutput
            Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = i

        End If
    End If

    'Now reset any areas without cellular coverage
    If (Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value < 0.0001) Then

        Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = 0

    End If

Next currY
Next currX
Next i

'Now Paint the area (always paint the first and last car)
If ((PaintArea = vbYes) Or (CurrentCar = NumCars) Or (CurrentCar = 1)) Then
For currX = 1 To 100
    For currY = 1 To 100
        'Paint the area
        If ((Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value + 35) > 56) Then

            Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Interior.ColorIndex =
            Worksheets("Tower Coverage").Cells(currX + 1, currY +
            1).Value + 2 'Restart if all colors are used

        Else

            Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Interior.ColorIndex =
            Worksheets("Tower Coverage").Cells(currX + 1, currY +
            1).Value + 35 'Begin with pastels

        End If

        'Paint the areas without cellular coverage white
        If (Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = 0) Then

            Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Interior.ColorIndex = 2 'White

        End If

    Next currY
Next currX

If (PaintArea = vbYes) Then
    Worksheets("Tower Coverage").Select
    Stop

```

```

End If

End If

If (TowerSetupOnly = vbYes) Then 'Don't iterate cars if TowerSetupOnly
    Exit Sub
End If

'Create a travelling car and place it on a random starting node
Seed = Int((DifferentPaths - 1 + 1) * Rnd() + 1) 'Random integer path number
CurrentState = Int((Active - 1 + 1) * Rnd() + Passive) 'Driver is randomly Active or Passive
XPos = NodesX(Seed, 1)
Sheets("Current Parameters").Range("C1").Value = XPos
YPos = NodesY(Seed, 1)
Sheets("Current Parameters").Range("C2").Value = YPos
CurrentNode = 1
NextNode = 2
FinalNode = 0

'Determine number of nodes in car's path
For i = 1 To 10
    If NodesX(Seed, i) = 0 Then
        FinalNode = i - 1
        Exit For
    End If
Next i

'Set car speed
Sheets("Current Parameters").Range("C4").Formula = "=NORMINV(" & Rnd() & "," & SpeedMean & "," &
SpeedSD & ")" 'Inverse Normal function
Speed = Sheets("Current Parameters").Range("C4").Value
'Car travels (XDirection * Speed * dt) in the X direction per iteration
'Car travels (YDirection * Speed * dt) in the Y direction per iteration

'Set BeaconInterval for Passive drivers
If (CurrentState = Passive) Then
    Sheets("Current Parameters").Range("C5").Formula = "=NORMINV(" & Rnd() & "," & BeaconIntervalMean
    & "," & BeaconIntervalSD & ")"
    BeaconInterval = Sheets("Current Parameters").Range("C5").Value
End If

'Set car direction
If (NodesX(Seed, 2) = XPos) Or (NodesY(Seed, 2) = YPos) Then 'Need to set in case of arctan(infinity)
    If (NodesX(Seed, 2) = XPos) Then
        YDirection = Sgn(NodesY(Seed, 2) - YPos)
        XDirection = 0
        Sheets("Current Parameters").Range("G3").Value = XDirection
        Sheets("Current Parameters").Range("G4").Value = YDirection
        Sheets("Current Parameters").Range("G2").Value = "N/A"
        CurrentTower = Worksheets("Tower Coverage").Cells(NodesX(Seed, 1) + 1, NodesY(Seed, 1) +
        1).Value 'Assign start tower
        Worksheets("Car Results").Cells(CurrentCar + 2, 6).Value = CurrentTower
        Worksheets("Car Results").Cells(CurrentCar + 2, 7).Value = -1 'Termination value
        Worksheets("Tower Results").Cells(CurrentCar + 2, 6).Value = CurrentTower
        Worksheets("Tower Results").Cells(CurrentCar + 2, 7).Value = -1 'Termination value
    End If

    If (NodesY(Seed, 2) = YPos) Then
        XDirection = Sgn(NodesX(Seed, 2) - XPos)
        YDirection = 0
        Sheets("Current Parameters").Range("G3").Value = XDirection
        Sheets("Current Parameters").Range("G4").Value = YDirection
        Sheets("Current Parameters").Range("G2").Value = "N/A"
        CurrentTower = Worksheets("Tower Coverage").Cells(NodesX(Seed, 1) + 1, NodesY(Seed, 1) +
        1).Value 'Assign start tower
        Worksheets("Car Results").Cells(CurrentCar + 2, 6).Value = CurrentTower
    End If
End If

```



```

'Check Passive tower dependency
'NOTE: Passive tower locations will overwrite active ones already written (this is correct)
If (CurrentState = Passive) Then

    If (BeaconInterval > ElapsedTime) Then

        Worksheets("Car Results").Cells(CurrentCar + 2, 9 + (OldTowers * 4)).Value = (Time +
            BeaconInterval - ElapsedTime)

        'Crossover location happens when passive user detects handover
        If (Round(XPos) > XPos) Then

            Worksheets("Tower Results").Cells(CurrentCar + 2, 8 + (OldTowers * 3)).Value =
                (XPos + 0.5) + (XDirection * Speed * (BeaconInterval - ElapsedTime))

        Else

            Worksheets("Tower Results").Cells(CurrentCar + 2, 8 + (OldTowers * 3)).Value =
                (XPos - 0.5) + (XDirection * Speed * (BeaconInterval - ElapsedTime))

        End If

        If (Round(YPos) > YPos) Then

            Worksheets("Tower Results").Cells(CurrentCar + 2, 9 + (OldTowers * 3)).Value =
                (YPos + 0.5) + (YDirection * Speed * (BeaconInterval - ElapsedTime))

        Else

            Worksheets("Tower Results").Cells(CurrentCar + 2, 9 + (OldTowers * 3)).Value =
                (YPos - 0.5) + (YDirection * Speed * (BeaconInterval - ElapsedTime))

        End If

    End If

    If (BeaconInterval <= ElapsedTime) Then

        Worksheets("Car Results").Cells(CurrentCar + 2, 9 + (OldTowers * 4)).Value = Time

    End If

    'Calculate difference in passive/active times
    Worksheets("Car Results").Cells(CurrentCar + 2, 10 + (OldTowers * 4)).Value =
        (Worksheets("Car Results").Cells(CurrentCar + 2, 9 + (OldTowers * 4)).Value - Time)

    End If

    ElapsedTime = 0
    OldTowers = OldTowers + 1

End If

XPos = XPos + (XDirection * Speed * dt) 'Move car in X direction
YPos = YPos + (YDirection * Speed * dt) 'Move car in Y direction
Time = Time + dt 'Increment time
ElapsedTime = ElapsedTime + dt

If (ElapsedTime > BeaconInterval) Then 'Make sure ElapsedTime does not
    ElapsedTime = 0 'exceed BeaconInterval

End If

If (Abs((XPos - NodesX(Seed, NextNode)) + (YPos - NodesY(Seed, NextNode))) < Precision) Then

    CurrentNode = CurrentNode + 1 'Increment last node visited
    NextNode = CurrentNode + 1 'Increment next node

    XPos = NodesX(Seed, CurrentNode) 'Recalibrate X position
    YPos = NodesY(Seed, CurrentNode) 'Recalibrate Y position

    'Reset Direction
    If (NodesX(Seed, NextNode) = XPos) Or (NodesY(Seed, NextNode) = YPos) Then 'Need to set in
        'case of arctan(infinity)

```

```

If (NodesX(Seed, NextNode) = XPos) Then

    YDirection = Sgn(NodesY(Seed, NextNode) - YPos)
    XDirection = 0
    Sheets("Current Parameters").Range("G3").Value = XDirection
    Sheets("Current Parameters").Range("G4").Value = YDirection
    Sheets("Current Parameters").Range("G2").Value = "N/A"

End If

If (NodesY(Seed, NextNode) = YPos) Then

    XDirection = Sgn(NodesX(Seed, NextNode) - XPos)
    YDirection = 0
    Sheets("Current Parameters").Range("G3").Value = XDirection
    Sheets("Current Parameters").Range("G4").Value = YDirection
    Sheets("Current Parameters").Range("G2").Value = "N/A"

End If

Else

    Sheets("Current Parameters").Range("G2").Formula = "-ATAN(" & ((XPos - NodesX(Seed,
        NextNode)) / (NodesY(Seed, NextNode) - YPos)) & ")"
    Sheets("Current Parameters").Range("G3").Formula = "=SIN(G2)"
    Sheets("Current Parameters").Range("G4").Formula = "=COS(G2)"
    XDirection = Sgn(NodesX(Seed, NextNode) - XPos) * (((Sheets("Current
        Parameters").Range("G3").Value) ^ 2) ^ (1 / 2))
        'Sign needs to be taken into account
    YDirection = Sgn(NodesY(Seed, NextNode) - YPos) * (((Sheets("Current
        Parameters").Range("G4").Value) ^ 2) ^ (1 / 2))
        'Same here
    Sheets("Current Parameters").Range("G3").Value = XDirection
    Sheets("Current Parameters").Range("G4").Value = YDirection

End If

End If

Wend

Sheets("Car Results").Select
Cells(CurrentCar + 2, 1).Value = CurrentCar
Cells(CurrentCar + 2, 2).Value = Speed
Cells(CurrentCar + 2, 3).Value = Seed
Cells(CurrentCar + 2, 4).Value = Time

If (CurrentState = Active) Then

    Cells(CurrentCar + 2, 5).Value = "Active"

End If

If (CurrentState = Passive) Then

    Cells(CurrentCar + 2, 5).Value = "Passive"
    Cells(CurrentCar + 2, 27).Value = BeaconInterval

End If

Sheets("Tower Results").Select
Cells(CurrentCar + 2, 1).Value = CurrentCar
Cells(CurrentCar + 2, 2).Value = Speed
Cells(CurrentCar + 2, 3).Value = Seed
Cells(CurrentCar + 2, 4).Value = Time

If (CurrentState = Active) Then

    Cells(CurrentCar + 2, 5).Value = "Active"

End If

If (CurrentState = Passive) Then

    Cells(CurrentCar + 2, 5).Value = "Passive"

End If

Time = 0

```

```

ElapsedTime = 0
OldTowers = 0

Next CurrentCar      'Iterate whole loop over all cars

'Guess paths if GuessPaths = vbYes
If (GuessPaths = vbYes) Then
  'Use TowerOutputMean to determine standard possibilities
  'First recalculate TowerCoverage sheet
  For i = 1 To DifferentTowers

    'Calculate tower strength
    TowerOutput = TowerOutputMean

    For currX = 1 To 100

      For currY = 1 To 100

        CurrentDistance = (((TowersX(i) - currX) ^ 2) + ((TowersY(i) - currY) ^ 2)) ^ (1 / 2)
                          + 1 'Assume that the car is not actually on a tower

        If (i = 1) Then      'Initialize all values to first tower then compare when i > 1
          If (CurrentDistance > 0) Then
            Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value = TowerOutput /
              (CurrentDistance ^ OrderOfDecay)

          End If

          If (CurrentDistance = 0) Then
            Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value = TowerOutput

          End If

          Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = 1

        End If

        If (i > 1) Then
          If (CurrentDistance > 0) Then 'Only replace if current tower is more powerful
            'than existing tower

            If ((TowerOutput / (CurrentDistance ^ OrderOfDecay)) > Worksheets("Tower
              Strength").Cells(currX + 1, currY + 1).Value) Then

              Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value =
                TowerOutput / (CurrentDistance ^ OrderOfDecay)
              Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = i

            End If

          End If

          If (CurrentDistance = 0) Then 'Obviously no other tower is stronger at these
            'coordinates

            Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value = TowerOutput
            Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = i

          End If

        End If

        'Now reset any areas without cellular coverage
        If (Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value < 0.0001) Then

          Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = 0

        End If

      Next currY

    Next currX

  Next i

```

```

'Now Paint the area
For currX = 1 To 100

  For currY = 1 To 100

    If ((Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value + 35) > 56) Then

      Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Interior.ColorIndex =
        Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value + 2
      'Restart if all colors are used

    Else

      Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Interior.ColorIndex =
        Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value + 35
      'Begin with pastels

    End If

    'Paint the areas without cellular coverage white
    If (Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = 0) Then

      Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Interior.ColorIndex = 2
      'White

    End If

  Next currY

Next currX

'Determine path behaviors
ReDim PathsConsidered(DifferentPaths, 10) 'Handle up to 9 handovers
Speed = SpeedMean
For i = 1 To DifferentPaths

  XPos = NodesX(i, 1)
  YPos = NodesY(i, 1)
  OldTowers = 0
  CurrentNode = 1
  NextNode = 2
  FinalNode = 0

  'Determine number of nodes in car's path
  For j = 1 To 10

    If NodesX(i, j) = 0 Then

      FinalNode = j - 1
      Exit For

    End If

  Next j

  'Set car direction
  If (NodesX(i, 2) = XPos) Or (NodesY(i, 2) = YPos) Then 'Need to set in case of
                                                         arctan(infinity)

    If (NodesX(i, 2) = XPos) Then

      YDirection = Sgn(NodesY(i, 2) - YPos)
      XDirection = 0
      CurrentTower = Worksheets("Tower Coverage").Cells(NodesX(i, 1) + 1, NodesY(i, 1) +
        1).Value 'Assign start tower

    End If

    If (NodesY(i, 2) = YPos) Then

      XDirection = Sgn(NodesX(i, 2) - XPos)
      YDirection = 0
      CurrentTower = Worksheets("Tower Coverage").Cells(NodesX(i, 1) + 1, NodesY(i, 1) +
        1).Value 'Assign start tower

    End If

  Else

    Sheets("Current Parameters").Range("G2").Formula = "=-ATAN(" & ((XPos - NodesX(i, 2)) /

```

```

                                                                    (NodesY(i, 2) - YPos) & ")")
Sheets("Current Parameters").Range("G3").Formula = "=SIN(G2)"
Sheets("Current Parameters").Range("G4").Formula = "=COS(G2)"
XDirection = Sgn(NodesX(i, 2) - XPos) * (((Sheets("Current Parameters").Range("G3").Value
                                                                    ^ 2) ^ (1 / 2)) 'Sign needs to be taken into account
YDirection = Sgn(NodesY(i, 2) - YPos) * (((Sheets("Current Parameters").Range("G4").Value
                                                                    ^ 2) ^ (1 / 2)) 'Same here
CurrentTower = Worksheets("Tower Coverage").Cells(NodesX(i, 1) + 1, NodesY(i, 1) +
                                                                    1).Value 'Assign start tower

End If

PathsConsidered(i, 1) = CurrentTower 'Assign start tower

'Move car along path
While (CurrentNode < FinalNode)
  'Check Active tower dependency
  If (Worksheets("Tower Coverage").Cells(Round(XPos) + 1, Round(YPos) + 1).Value <=
                                                                    CurrentTower) Then 'NOTE: Round is less precise than
                                                                    'Precision, so the last tower will ALWAYS be recorded
    CurrentTower = Worksheets("Tower Coverage").Cells(Round(XPos) + 1, Round(YPos) +
                                                                    1).Value
    If (OldTowers < 8) Then
      PathsConsidered(i, 2 + OldTowers) = CurrentTower
      OldTowers = OldTowers + 1
    End If
  End If
End If

XPos = XPos + (XDirection * Speed * dt) 'Move car in X direction
YPos = YPos + (YDirection * Speed * dt) 'Move car in Y direction

If (Abs((XPos - NodesX(i, NextNode)) + (YPos - NodesY(i, NextNode))) < Precision) Then

  CurrentNode = CurrentNode + 1 'Increment last node visited
  NextNode = CurrentNode + 1 'Increment next node

  XPos = NodesX(i, CurrentNode) 'Recalibrate X position
  YPos = NodesY(i, CurrentNode) 'Recalibrate Y position

  'Reset Direction
  If (NodesX(i, NextNode) = XPos) Or (NodesY(i, NextNode) = YPos) Then 'Need to set in
                                                                    'case of arctan(infinity)
    If (NodesX(i, NextNode) = XPos) Then
      YDirection = Sgn(NodesY(i, NextNode) - YPos)
      XDirection = 0
    End If
    If (NodesY(i, NextNode) = YPos) Then
      XDirection = Sgn(NodesX(i, NextNode) - XPos)
      YDirection = 0
    End If
  Else
    Sheets("Current Parameters").Range("G2").Formula = "=ATAN(" & ((XPos - NodesX(i,
                                                                    NextNode)) / (NodesY(i, NextNode) - YPos)) & ")")
    Sheets("Current Parameters").Range("G3").Formula = "=SIN(G2)"
    Sheets("Current Parameters").Range("G4").Formula = "=COS(G2)"
    XDirection = Sgn(NodesX(i, NextNode) - XPos) * (((Sheets("Current
    Parameters").Range("G3").Value ^ 2) ^ (1 / 2)) 'Sign needs to be taken into account
    YDirection = Sgn(NodesY(i, NextNode) - YPos) * (((Sheets("Current
    Parameters").Range("G4").Value ^ 2) ^ (1 / 2)) 'Same here
  End If
End If

Wend

PathsConsidered(i, 2 + OldTowers) = -1 'Termination value

Next i

```

```

'Iterate over each car index and guess path
For i = 1 To NumCars
    'Initialize FeasibleSet
    For j = 1 To 9
        FeasibleSet(j) = 1 'Path is feasible until proven otherwise
    Next j
    'Path is not feasible if index is > DifferentPaths
    For j = 1 To 9
        If (j > DifferentPaths) Then
            FeasibleSet(j) = 0
        End If
    Next j
    'Check starting tower
    CurrentEvaluation = Worksheets("Car Results").Cells(2 + i, 6).Value
    For j = 1 To DifferentPaths
        If (CurrentEvaluation <> PathsConsidered(j, 1)) Then
            FeasibleSet(j) = 0
        End If
    Next j
    k = 2 'Reset k index
    CurrentEvaluation = Worksheets("Car Results").Cells(2 + i, 7).Value
    'Check remaining towers
    While (CurrentEvaluation > 0) 'Termination value of -1 will exit loop
        For j = 1 To DifferentPaths
            If (CurrentEvaluation <> PathsConsidered(j, k)) Then
                FeasibleSet(j) = 0
            End If
        Next j
        k = k + 1 'Increment k
        CurrentEvaluation = Worksheets("Car Results").Cells(2 + i, 7 + (4 * (k - 2))).Value
    Wend
    'Display results in "Car Results" sheet
    For j = 1 To 9
        Worksheets("Car Results").Cells(2 + i, 27 + j).Value = FeasibleSet(j)
    Next j
Next i
'Replace values if soft estimation is used (this is ok because path guessing is very quick)
If (SoftEstimation = vbYes) Then
    Worksheets("Car Results").Select
    Range(Cells(3, 28), Cells(NumCars + 2, 36)).Select
    Selection.Clear
    For i = 1 To NumCars
        For j = 1 To 9
            FeasibleSet(j) = 1 'Path is feasible until proven otherwise
            If (j > DifferentPaths) Then

```

```

        FeasibleSet(j) = 0
    End If
Next j

'Determine how many handovers there are in SoftPath
k = 1
CurrentEvaluation = Worksheets("Car Results").Cells(2 + i, 7).Value

While (CurrentEvaluation > 0)

    k = k + 1
    CurrentEvaluation = Worksheets("Car Results").Cells(2 + i, 7 + (4 * (k - 1))).Value

Wend

SoftPathLength = k
ReDim SoftPath(SoftPathLength) 'There are k towers to be stored in SoftPath

'Initialize SoftPath
SoftPath(1) = Worksheets("Car Results").Cells(2 + i, 6).Value
If (k > 1) Then

    For j = 2 To SoftPathLength

        SoftPath(j) = Worksheets("Car Results").Cells(2 + i, 7 + (4 * (j - 2))).Value

    Next j

End If

'Now check SoftPath against available paths
For j = 1 To DifferentPaths

    'Determine length of each path in PathsConsidered
    CurrentPathLength = 1
    CurrentEvaluation = PathsConsidered(j, 2)

    While (CurrentEvaluation > 0)

        CurrentPathLength = CurrentPathLength + 1
        CurrentEvaluation = PathsConsidered(j, CurrentPathLength + 1)

    Wend

    If (SoftPathLength > CurrentPathLength) Then

        FeasibleSet(j) = 0

    End If

    'Only look for SoftPath if eligible
    If (SoftPathLength <= CurrentPathLength) Then

        SoftPathTries = (CurrentPathLength - SoftPathLength + 1)

        'Search for SoftPath
        Offset = 0
        k = 1
        Found = False
        While (Offset < SoftPathTries)

            If SoftPath(k) = PathsConsidered(j, k + Offset) Then

                k = k + 1
                If (k = (SoftPathLength + 1)) Then

                    Found = True
                    Offset = SoftPathTries 'Exit loop

                End If

            Else

                Offset = Offset + 1
                k = 1

            End If

        End While

    End If

End For

```

```

        End If
    Wend

    'Record result
    If (Found = False) Then

        FeasibleSet(j) = 0

    End If
End If

Next j

'Display results
For j = 1 To 9

    Worksheets("Car Results").Cells(2 + i, 27 + j).Value = FeasibleSet(j)

Next j

Next i

End If 'Soft estimation subroutine
End If 'Path guessing subroutine

'Format results
Worksheets("Car Results").Select
Range(Cells(3, 1), Cells(NumCars + 3, 1)).Select
With Selection
    .VerticalAlignment = xlBottom
    .WrapText = False
    .Orientation = 0
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
    .Font.Bold = True
End With

'Hide -1 (termination) values
For i = 0 To NumCars

    For j = 0 To 4

        If Worksheets("Car Results").Cells(i + 3, (j * 4) + 7).Value = -1 Then

            Worksheets("Car Results").Cells(i + 3, (j * 4) + 7).Select
            With Selection.Font
                .Name = "Arial"
                .FontStyle = "Regular"
                .Size = 10
                .Strikethrough = False
                .Superscript = False
                .Subscript = False
                .OutlineFont = False
                .Shadow = False
                .Underline = xlUnderlineStyleNone
                .ColorIndex = 2 'White to hide -1 (termination) values

            End With

        End If

    Next j

Next i

Range(Cells(3, 1), Cells(NumCars + 3, 36)).Select
Selection.HorizontalAlignment = xlCenter

Worksheets("Tower Results").Select
Range(Cells(3, 1), Cells(NumCars + 3, 1)).Select
With Selection
    .VerticalAlignment = xlBottom

```

```
.WrapText = False
.Orientation = 0
.AddIndent = False
.IndentLevel = 0
.ShrinkToFit = False
.ReadingOrder = xlContext
.MergeCells = False
.Font.Bold = True
End With
'Hide -1 (termination) values
For i = 0 To NumCars
    For j = 0 To 4
        If Worksheets("Tower Results").Cells(i + 3, (j * 3) + 7).Value = -1 Then
            Worksheets("Tower Results").Cells(i + 3, (j * 3) + 7).Select
            With Selection.Font
                .Name = "Arial"
                .FontStyle = "Regular"
                .Size = 10
                .Strikethrough = False
                .Superscript = False
                .Subscript = False
                .OutlineFont = False
                .Shadow = False
                .Underline = xlUnderlineStyleNone
                .ColorIndex = 2    'White to hide -1 (termination) values
            End With
        End If
    Next j
Next i
Range(Cells(3, 1), Cells(NumCars + 3, 36)).Select
Selection.HorizontalAlignment = xlCenter
Cells(3, 1).Select
Worksheets("Car Results").Select
Cells(3, 1).Select
End Sub
```

Procedure ResetMaps

```

Sub ResetMaps()
Dim ans As Integer

ans = MsgBox("Reset All Maps and Car Results Data?", vbYesNo)

If ans = vbYes Then
    Sheets("Tower Strength").Select
    Range("B2").Select
    For currY = 1 To 100
        For currX = 1 To 100
            Worksheets("Tower Strength").Cells(currX + 1, currY + 1).Value = 0
            Worksheets("Paths").Cells(currX + 1, currY + 1).Value = 0
            Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Value = 0
            Worksheets("Tower Coverage").Cells(currX + 1, currY + 1).Interior.ColorIndex = 2
        Next currX
    Next currY

    Worksheets("Tower Coverage").Select
    ActiveSheet.Shapes.SelectAll
    Selection.Delete

    Worksheets("Car Results").Select
    Range("A3:AA50000").Select
    Selection.Clear

    Worksheets("Tower Results").Select
    Range("A3:AA50000").Select
    Selection.Clear

    Worksheets("Paths").Select
    ActiveSheet.Shapes.SelectAll
    Selection.Delete

    'Clean up area
    Worksheets("Tower Coverage").Select
    Range("B2:CW101").Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .Weight = xlThick
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .Weight = xlThick
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlThick
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .Weight = xlThick
        .ColorIndex = xlAutomatic
    End With
    Selection.Borders(xlInsideVertical).LineStyle = xlNone
    Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
    Range("B2").Select

    Worksheets("Paths").Select
    Range("B2:CW101").Select
    With Selection
        .HorizontalAlignment = xlCenter

```

```
.VerticalAlignment = xlCenter
.WrapText = False
.Orientation = 0
.AddIndent = False
.IndentLevel = 0
.ShrinkToFit = False
.ReadingOrder = xlContext
.MergeCells = False
End With
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
.LineStyle = xlContinuous
.Weight = xlThick
.ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
.LineStyle = xlContinuous
.Weight = xlThick
.ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
.LineStyle = xlContinuous
.Weight = xlThick
.ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
.LineStyle = xlContinuous
.Weight = xlThick
.ColorIndex = xlAutomatic
End With
Selection.Borders(xlInsideVertical).LineStyle = xlNone
Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
Range("B2").Select
End If
End Sub
```

Procedure TowerLocator

```

Sub TowerLocator()
Dim NumCars, i, j As Integer
Dim KeepGoing As Boolean
Dim TempX, TempY As Double
Dim CurrentGuessX, CurrentGuessY As Double
Dim Precision As Double
Dim CurrentXError, CurrentYError, TotalXError, TotalYError As Double
Dim ActualTowerLocationX, ActualTowerLocationY As Integer

'Clear existing sheet
Worksheets("Tower Location Guessing").Select
Range("A3:AA50000").Select
Selection.Clear

'Initialize variables
NumCars = 1
KeepGoing = True
TempX = 0
TempY = 0
Precision = 0.1
CurrentXError = 0
CurrentYError = 0
TotalXError = 0
TotalYError = 0

If (Worksheets("Tower Results").Cells(3, 1).Value <> 1) Then
    MsgBox "No Tower Results!"
    Exit Sub
End If

While (KeepGoing = True)
    If (Worksheets("Tower Results").Cells(3 + NumCars, 1).Value >= NumCars) Then
        NumCars = NumCars + 1
    Else
        KeepGoing = False
    End If
Wend

For i = 1 To NumCars
    Worksheets("Tower Location Guessing").Cells(2 + i, 1).Value = Worksheets("Tower Results").Cells(2
        + i, 1).Value
    Worksheets("Tower Location Guessing").Cells(2 + i, 2).Value = Worksheets("Tower Results").Cells(2
        + i, 8).Value
    Worksheets("Tower Location Guessing").Cells(2 + i, 3).Value = Worksheets("Tower Results").Cells(2
        + i, 9).Value
    Worksheets("Tower Location Guessing").Cells(2 + i, 4).Value = Worksheets("Tower Results").Cells(2
        + i, 11).Value
    Worksheets("Tower Location Guessing").Cells(2 + i, 5).Value = Worksheets("Tower Results").Cells(2
        + i, 12).Value
Next i

Worksheets("Tower Location Guessing").Select
'Initialize TotalError
For i = 1 To NumCars
    TotalXError = TotalXError + ((Cells(2 + i, 2)) ^ 2)
    TotalYError = TotalYError + ((Cells(2 + i, 3)) ^ 2)
    TotalXError = TotalXError + ((Cells(2 + i, 4)) ^ 2)
    TotalYError = TotalYError + ((Cells(2 + i, 5)) ^ 2)
Next i

'Guess tower location
While (TempX < 100)
    While (TempY < 100)
        CurrentYError = 0

        For i = 1 To NumCars
            CurrentYError = CurrentYError + ((Cells(2 + i, 3) - TempY) ^ 2)
            CurrentYError = CurrentYError + ((Cells(2 + i, 5) - TempY) ^ 2)
        Next i

        If (CurrentYError < TotalYError) Then
            TotalYError = CurrentYError
            CurrentGuessY = TempY
        End If
        TempY = TempY + Precision
    Wend
Wend

```

```

CurrentXError = 0
For i = 1 To NumCars
    CurrentXError = CurrentXError + ((Cells(2 + i, 2) - TempX) ^ 2)
    CurrentXError = CurrentXError + ((Cells(2 + i, 4) - TempX) ^ 2)
Next i

If (CurrentXError < TotalXError) Then
    TotalXError = CurrentXError
    CurrentGuessX = TempX
End If

TempX = TempX + Precision
Wend

Cells(3, 7).Value = CurrentGuessX
Cells(3, 8).Value = CurrentGuessY
Cells(3, 9).Value = TotalXError
Cells(3, 10).Value = TotalYError

'Locate actual tower
For i = 1 To 100
    For j = 1 To 100
        If (Worksheets("Paths").Cells(i + 1, j + 1).Value = 101) Then
            ActualTowerLocationX = i
            ActualTowerLocationY = j
        End If
    Next j
Next i

'Calculate total error
Worksheets("Tower Location Guessing").Cells(3, 11).Value = Worksheets("Tower Location
Guessing").Cells(3, 9).Value + Worksheets("Tower Location Guessing").Cells(3, 10).Value

'Format results
Range(Cells(3, 1), Cells(NumCars + 3, 1)).Select
With Selection
    .VerticalAlignment = xlBottom
    .WrapText = False
    .Orientation = 0
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
    .Font.Bold = True
End With

Range(Cells(3, 1), Cells(NumCars + 3, 5)).Select
Selection.HorizontalAlignment = xlCenter

Range(Cells(3, 7), Cells(3, 13)).Select
With Selection
    .VerticalAlignment = xlBottom
    .WrapText = False
    .Orientation = 0
    .AddIndent = False
    .IndentLevel = 0
    .ShrinkToFit = False
    .ReadingOrder = xlContext
    .MergeCells = False
    .Font.Bold = True
    .HorizontalAlignment = xlCenter
End With

Worksheets("Tower Location Guessing").Cells(3, 12).Value = ActualTowerLocationX
Worksheets("Tower Location Guessing").Cells(3, 13).Value = ActualTowerLocationY

Cells(3, 1).Select

End Sub

```

Procedure MonteCarlo

```
Sub MonteCarlo()
Randomize
Dim i, Times, FirstIndex As Integer
Dim Sum, Avg As Double
Dim p As Double
Dim Found As Boolean

For p = 1 To 100
    Found = False
    Avg = 0
    Sum = 0
    Worksheets("Data").Cells(1, 3) = (p / 100)

    For Times = 1 To 1000
        Calculate
        For i = 1 To 200
            If ((Cells(3 + i, 3) = 0) And (Found = False)) Then
                FirstIndex = i
                Found = True
            End If
        Next i

        Sum = Sum + FirstIndex
        Avg = Sum / Times

        Found = False
    Next Times

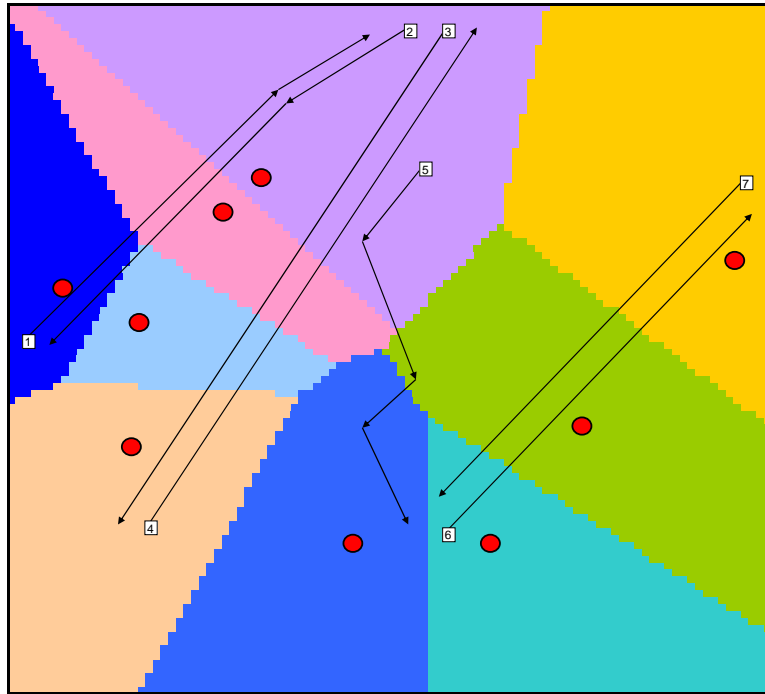
    Worksheets("Data").Cells(1, 7).Value = Avg
    Worksheets("Graph").Cells(p, 1).Value = p
    Worksheets("Graph").Cells(p, 2).Value = Avg
Next p
End Sub
```

Mobiledia Tower Search HTML Code

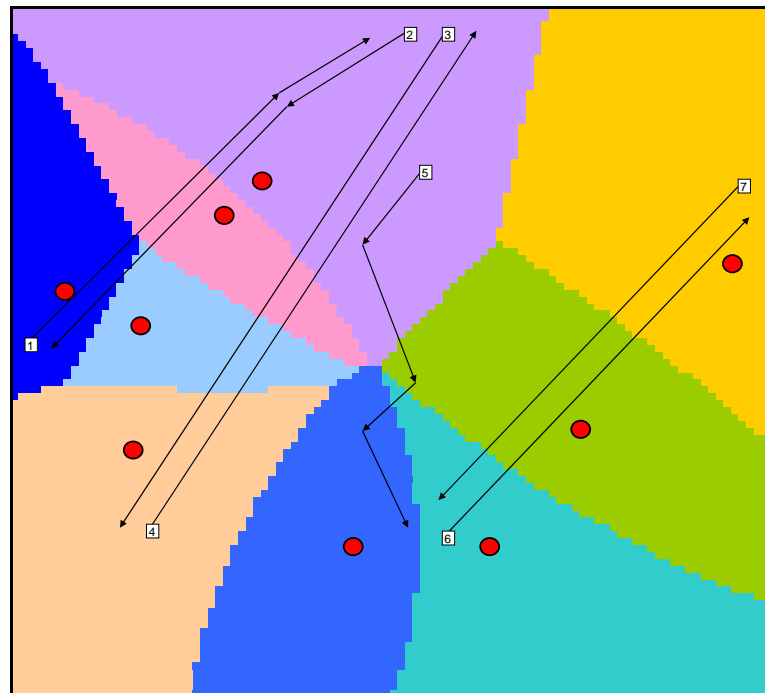
```

<html>
<table border="1" cellpadding="5" width="125" bordercolor="#C0C0C0" cellspacing="0">
  <tr>
    <td align="center" bgcolor="#5D7895"><b>
      <a style="text-decoration:none" href="http://www.cellreception.com">
        <font face="Verdana" size="2" color="#FFFFFF">Cell Phone Tower Search</font></a></b><br>
      </td>
  </tr>
  <tr>
    <td align="right" bgcolor="#FFFFFF"><br>
      <form method="post" action="http://www.cellreception.com/towers/towers.php">
        <input type="text" onfocus="this.value=''" size="14" value="City" name="city"><br>
        <select size="1" name="state_abr">
          <option value="0" selected>--</option>
          <option value="AK">AK</option>
          <option value="AL">AL</option>
          <option value="AR">AR</option>
          <option value="AZ">AZ</option>
          <option value="CA">CA</option>
          <option value="CO">CO</option>
          <option value="CT">CT</option>
          <option value="DC">DC</option>
          <option value="DE">DE</option>
          <option value="GA">GA</option>
          <option value="FL">FL</option>
          <option value="HI">HI</option>
          <option value="IA">IA</option>
          <option value="ID">ID</option>
          <option value="IL">IL</option>
          <option value="IN">IN</option>
          <option value="KS">KS</option>
          <option value="KY">KY</option>
          <option value="LA">LA</option>
          <option value="MA">MA</option>
          <option value="MD">MD</option>
          <option value="ME">ME</option>
          <option value="MI">MI</option>
          <option value="MN">MN</option>
          <option value="MO">MO</option>
          <option value="MS">MS</option>
          <option value="MT">MT</option>
          <option value="NC">NC</option>
          <option value="ND">ND</option>
          <option value="NE">NE</option>
          <option value="NH">NH</option>
          <option value="NJ">NJ</option>
          <option value="NM">NM</option>
          <option value="NV">NV</option>
          <option value="NY">NY</option>
          <option value="OH">OH</option>
          <option value="OK">OK</option>
          <option value="OR">OR</option>
          <option value="PA">PA</option>
          <option value="RI">RI</option>
          <option value="SC">SC</option>
          <option value="SD">SD</option>
          <option value="TN">TN</option>
          <option value="TX">TX</option>
          <option value="UT">UT</option>
          <option value="VT">VT</option>
          <option value="VA">VA</option>
          <option value="WA">WA</option>
          <option value="WI">WI</option>
          <option value="WV">WV</option>
          <option value="WY">WY</option>
        </select>
        <input type="submit" value="Go" name="go"></form>
        <a href="http://www.mobiledia.com">
          <p align="right">
            </a><br>
            <font face="Verdana" size="1">A <a style="text-decoration:none" href="http://www.mobiledia.com">
              <font color="#000000">Cell Phone</font></a> Resource Site</font></p></td>
  </tr>
</table>
</html>

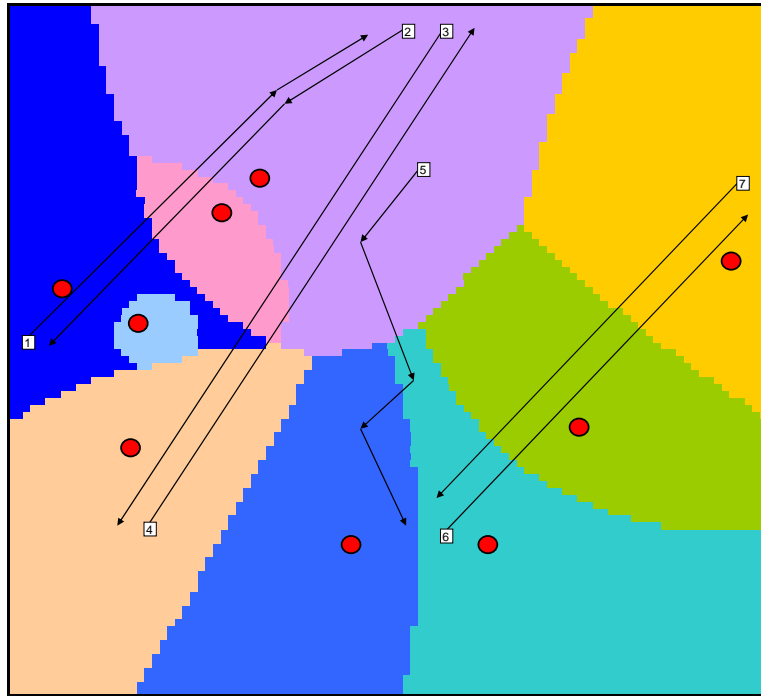
```

SimulateSubscriber output under different σ values

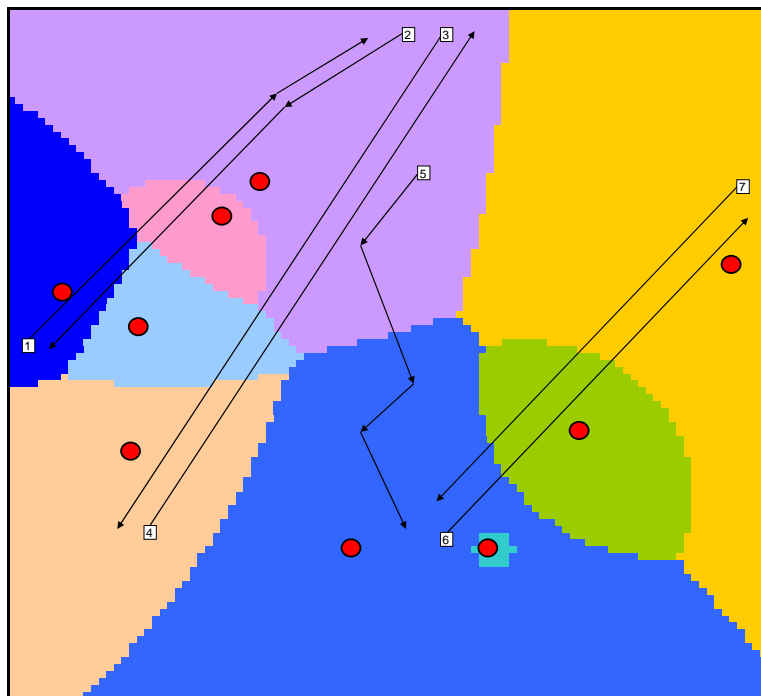
$$P_0 \sim N(\mu, \sigma), \sigma = 0$$



$$P_0 \sim N(\mu, \sigma), \sigma = \frac{\mu}{3}$$



$$P_0 \sim N(\mu, \sigma), \sigma = \frac{\mu}{2}$$



$$P_0 \sim N(\mu, \sigma), \sigma = \mu$$