Autonomous 802.11b Antenna Alignment using Linux

Author: Jason Olkonen
Northern Michigan University
Network Computing Undergraduate
CS 480 Senior Project
2003 Winter Semester
Date: 4/21/03

The purpose of this paper is to document the construction, use and applications of the autonomous 802.11b antenna alignment system developed for my senior project in Network Computing at Northern Michigan University.

I originally started thinking about this project during the summer of 2002 while working for NMU Telecom. Part of my job involved installing wireless networking equipment on and around the university campus, while working I realized the lack of precision involved with the current methods of antenna alignment. We would have two people communicate via handheld radios and move antennas around to get the best signal out of the equipment. The problem with this system is moving the antennas back to the best spot after shifting them around a for a half an hour with nothing to record the position of the antennas. This normally gave decent results but I knew that in some installations the signal could be improved, but time and lack of a good way to record the azimuth, elevation and signal strength in one package prevented this. This is where the idea for this project came about.

I started looking around for information on the subject but could find nobody that was in the same situation as I was. This led me to find various pieces of software and libraries that would do part of what I wanted to do. After acquiring the software, I began

to read the source code, which showed me what could be done with the hardware I had available. My next task became assembling the hardware, this proved more difficult than I had imagined. My first problem was USB support on the laptop. It turns out that the USB controller on the NM-1 laptop was not supported by the Linux kernel. After a week or two of trying to get the current kernel to work with the controller I put the project on hold. However, about two weeks after that and many searches on the internet for help it finally came in the 2.4.21-pre4 pre-patch to the 2.4.20 kernel. Yes, coincidentally somebody made the exact driver I needed, what luck! The luck did not flow into both of my USB to serial adaptors though. I managed to get one working (a Xircom Port Gear) on the NM-1 laptop, but neither the NM-1 nor the NU-4 laptops would work with the Belkin adaptor. The next pieces of hardware I tackled were the servos and their controllers. I managed to acquire five servos and two servos controllers from the Hexapod, which I believe was a previous senior project. Both of the servos controllers worked perfectly, unfortunately not all of the servos did. Out of the five servos I got two working and threes sub par ones. One continually rotated once it got past a certain point and never stopped while the other two moved so slowly that they were of little use to this project. Actually, the Mini SSC servo controllers did have a problem, straight from the factory. The problem is the unidirectional communication between the computer and the controller. This presents problems when moving the servos large distances because there is no way to communicate back to the computer that it got to the place you asked it to go. It is for this reason that I had to call sleep() or usleep() whenever I moved the servos, this slows down the searching but prevents recording signal information before the servos have gotten to the right place. Therefore, with these setbacks (bad servos and flakey USB to serial adaptor support) I began to accept the reality that my idea of having two

antennas aligning themselves autonomously would not in fact become reality. Instead, I chose to use a Cisco wireless access point as the other end of the wireless link. This is actually a decent way to test a network anyway since it is the most common application for a wireless network, a client attaching to an access point.

Algorithms

There are three searching algorithms incorporated into this project. For reference, I tried to scan the whole area (140 degrees of azimuth by 60 degrees of elevation) in 1x1 degree increments, but this took far too long for my small battery pack and me to complete so I cut it off. Before I cut it off, I measured the time one azimuth sweep took (95 seconds), multiplied it by 60 degrees of elevation, and came up with 5700 seconds (1 hour and 35 minutes) for a full scan had it actually completed. The first algorithm I implemented searches the area in a five-degree increment both ways (I labeled it a 5x5 scan in the program). It starts the antenna at -70 degrees azimuth and -30 degrees elevation then moves the antenna 5 degrees to the right until it reaches +70 degrees. It then increments the elevation by 5 degrees and scans from right to left. This increase in elevation and swapping of azimuth directions continues until the elevation reaches +30 degrees. At each increment of the scan the link quality, signal strength in dBm and the noise level in dBm is sampled 100 times to account for any anomalies and recorded to arrays. This scan records 377 data points in 215 seconds (2 minutes 30 seconds). The program then searches the through the signal strength and determines the best alignment for the antenna. During the search process the azimuth, elevation, quality, signal level, and noise level are recorded in a comma separated value file (stats.csv) so that the results can be looked at in a spreadsheet program like Microsoft Excel. The link quality and

noise level were not taken in to consideration when determining the best alignment, because I found that quality was directly related to the signal level and the noise level never varied more than one dBm. The next algorithm (patch scan) was based on the previous but was designed to search the area much faster. It starts out like the previous algorithm at a -70 degree azimuth and -30 degree elevation but does a preliminary scan of the 140 by 60 degree area in 20 degree steps both ways. This produces 32 data points for the initial scan. This is enough to start because of the antennas high degree of directionality, see figure 1.

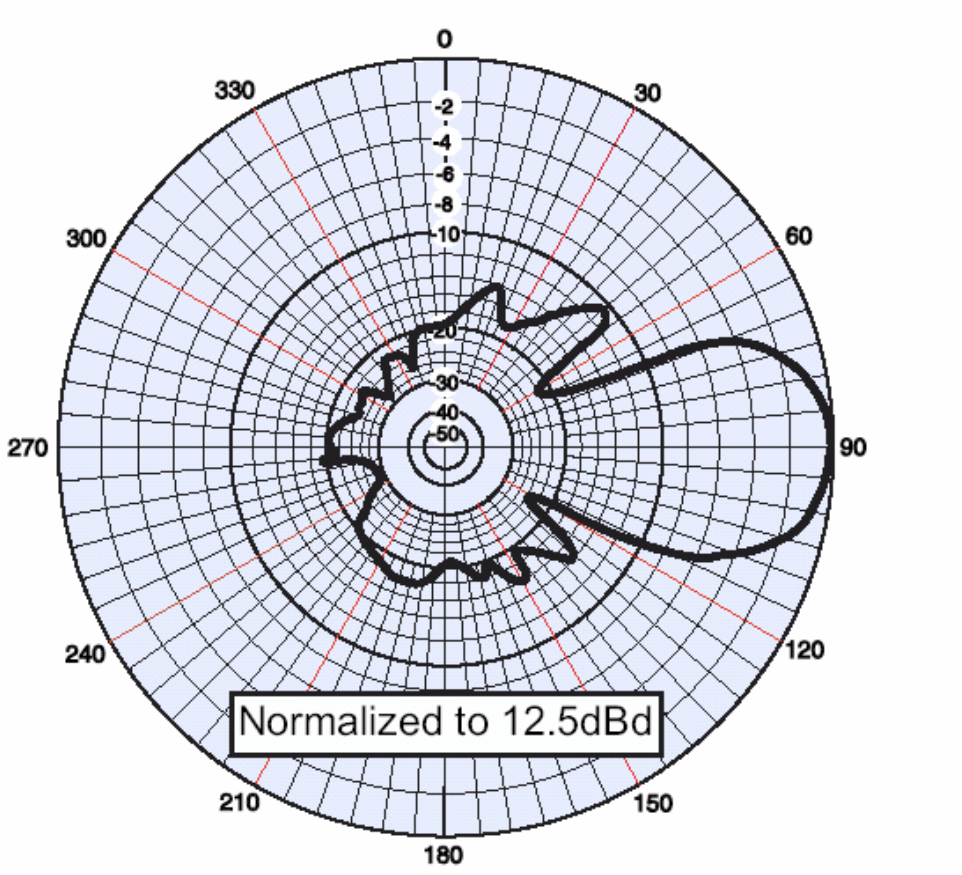

Fig 1. Vertical Field Pattern for the Antenex YE240015

From these 32 points, the best is chosen and a 10 by 10 area on all sides in scanned in a 5 degree increment. This produces results close to the first algorithm in only 27 seconds,

almost 8 times the speed of the 5x5 scan and 211 times the speed of the 1x1 scan. The next algorithm I tried is based on randomness; I thought I should see if random luck would find a good alignment for the antenna in a reasonable amount of time. For this, it loops through 30 random points, varying in azimuth and elevation, and finds the best signal out of it. The scan takes 66 seconds to complete, 3.25 times faster than the 5x5, 2.4 times slower than the patch scan, and 86 times faster than the 1x1 scan. The results from this scan were surprisingly good.

RESULTS

I tested each algorithm ten times on the same target and compared the results, see table 1. The results are in dBm, for which every 3 dB difference is a doubling of the received signal strength, dB = 10 x log10(power out/ power in). I consider the slow 5x5 scan the winner for reliability and best signal overall with -31 dBm ten times in a row. The patch scan and random scan each found a signal equal to or better than the 5x5 scan 3 times. The patch scan and random scan found the same signal level six times. Finally, the random scan outperformed the patch scan four times, which was a surprise to me.

	5x5	patch	random
	-31	-34	-34
	-31	-34	-34
	-31	-34	-34
	-31	-34	-33
	-31	-33	-32
	-31	-32	-32
	-31	-32	-32
	-31	-31	-31
	-31	-31	-30
	-31	-30	-29
Averages	-31	-32.5	-32.1

Table 1. Results of ten tests of each algorithm

USE OF THE SYSTEM

The system is very easy to use and requires minimal setup.

1) Connect the USB to serial adaptor to the servo controller.
2) Insert wireless card into computer.
3) Attach small coaxial cable to antenna.
4) Turn on computer
5) Login
6) Turn on power to servo controller
7) Type ./Align

```
Press the following keys:
The arrow keys move the antenna
F1 starts a 5x5 Scan
F2 starts a Patch Scan
F3 starts a Random Scan
F12 clears all of the stats from ram
b finds the best signal out of the positions tried
c centers the antenna
q quits the program
```

Example of a finished scan

```
Press the following keys:
The arrow keys move the antenna
F1 starts a 5x5 Scan
F2 starts a Patch Scan
F3 starts a Random Scan
F12 clears all of the stats from ram
b finds the best signal out of the positions tried
c centers the antenna
q quits the program


Azimuth: -2
Elevation: 0
BitRate:=11Mb/s
Link Quality | Signal level in dBm | Noise level in dBm
      64              -36                  -101
The best signal was found at azimuth:-2  and elevation:0 with a signal strength of -32 dBm
The same signal strength was found at 0 other spot(s)
```

SOFTWARE

Various pieces of software were incorporated into my project. Red Hat Linux, the Gnu g++ compiler, the Linux wireless extensions library, a C++ library for the Mini SSC-II Controller and the Ncurses library.

EQUIPMENT

My equipment was as follows: two Antenex Yagi 12.5dB antennas, one Cisco 350 Series Access point, two TS-80 servos, one Mini SSC II servo controller, various Fischer Technik parts for the antenna base with the servos, a base from a Dish networks satellite dish, a Lucent 802.11b PCMCIA card, a "pig tail" for the wireless card to the yagi antenna, Xircom USB to serial adapter, one IBM NM-1 Laptop computer, double sided tape, a bunch of zip ties, and electrical tape.

LINKS TO TECHNOLOGIES USED

Red Hat Linux - http://www.redhat.com/

Linux Kernel Source - http://www.kernel.org/

Wireless Extensions for Linux - http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html

NCURSES Programming HOWTO - http://en.tldp.org/HOWTO/NCURSES-Programming-HOWTO/

Scott Edwards Electronics Inc. (Mini SSC II servo controller) - http://www.seetron.com/ssc.htm

C++ software class library for the Mini SSC-II Controller - http://ece.clemson.edu/crb/students/vilas/professional/projects/MiniSSC/miniSSC.htm

Tower Hobbies (TS-80 Servo) - http://www2.towerhobbies.com/cgi-bin/wti0001p?&I=LXLN94&P=7

Antenex (12.5 dB Yagi antenna) - http://www.antenex.com/

Proxim ORiNOCO PC Card (the new name of the Lucent WaveLan 802.11b card) - http://www.proxim.com/products/all/orinoco/client/pccard/index.html

Cisco Aironet 350 Series Access Point - http://www.cisco.com/en/US/products/hw/wireless/ps458/ps447/index.html

Fischer Technik (antenna base) - http://www.fischertechnik.de/english/index.html