An Empirical Analysis of TinyOS RF Networking (and Beyond…)

Scott Klemmer, Sarah Waterson, and Kamin Whitehouse
CS Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720-1776 USA
+1 510 642 4948
{srk, waterson, kamin}@cs.berkeley.edu • http://guir.berkeley.edu/location

ABSTRACT
Context, particularly location, is an important source of information for human-computer interaction. In our project, we examine hardware, networking, and systems issues for a location sensing infrastructure. We present a thorough empirical analysis of the TinyOS RF motes. This analysis is leveraged to build a model of the RF signal strength. We have built a prototype system that employs a Kalman filter to determine distance between pairs of devices. Using a mass-spring system, we aggregate the distance measurements of all the networked devices. The location information is displayed with a visual user interface.

INTRODUCTION
The context of an application refers to information that is part of the application's operating environment. Typically, this includes information such as location, identity, activity and state of people, groups and objects [8] [12]. Of major interest are context-aware applications, which sense context information and modify their behavior accordingly without explicit user intervention. For example, an application that can capture a context and use it for later access would be able to provide context-based retrieval of stored information A location aware infrastructure opens up the space of possible context-enabled applications, such as a context aware tour guide (e.g., [1] [4], Figure 1) or instant messaging systems that can recognize when a user is at his or her desk versus wandering about his office.

While the HCI research community has done excellent work on applications, the underlying infrastructure support for these applications limited growth in this field. In order to fully realize a vision of ubiquitous computing [23], location sensing technologies must be small, wirelessly connected, and have long battery life.

RELATED WORK
A number of approaches have been used to determine location. GPS [7] and mobile phone Enhanced 911 [10] are technologies that focus on outdoor, wide range (~100m accuracy) location identification. For finer grain, indoor identification, RFID tags have been one of the more popular methods to date. One of the first successful location identification techniques is the Active Badges system pioneered at Olivetti Research [20] (related projects are [21] [19]). This system uses tags with infrared beacons that uniquely identify the wearer. Fixed sensors throughout the building collect and combine the information which is then polled by a central network controller. A large number of the badges were deployed and used for applications such as determining the probability of a person being in a specific place and the re-routing phone calls and messages to that location. While successful at many levels, the applications were limited by the infrared technology, which had considerable interference and only allowed one-way communication. No distance information is available from an infrared network, only connectivity. Bats [22] combine RF and ultrasound to quite accurately determine location in three dimensions, as well as the orientation of the device. A large number of fixed sensors, however, are required to support the system. Crickets [17], developed at MIT use a very similar combination of many sensors and RF with ultrasound to get room level granularity location determination.

A number of commercially available systems offer RF tagging technology for industrial applications (e.g. [2]); others such as Aetherwire are exploring ultra-wideband solutions to developing somewhat small (currently 2.0" x 3.6" x 1.5"), low-powered devices [3]. It is also possible to infer some location information from wireless networks such as Bluetooth [5].

Some current virtual reality tracking systems (e.g., [24] [11]) offer incredibly high tracking accuracy (up to 0.5 mm) in rooms (up to 4.5m x 8.5m) at interactive rates. These systems achieve this high level of performance at very high cost (often more than $10,000) to track a small number of objects (1-5), with a setup and calibration process that is inflexible and very labor intensive.

HARDWARE AND TINYOS
The TinyOS network sensors [13] [6]are small (1.5" x 1.5" x 0.5" including battery). The sensors are composed of a 4 MHz Atmel AVR 8535 processor with 8 kilobytes of flash as program memory and 512 bytes of SRAM as data memory. They have a 916.50 MHz CSMA RF wireless transceiver operating at up to 19.2 Kbps and have three LED's for output. At peak level, the hardware consumes 19.5 mA, running about 30 hours on a battery. In inactive mode, the system draws only 10 μA of current, enabling the battery to last for more than a year.

Figure 1. A Location-aware electronic guidebook [4]. In this system, location is ascertained by infrared connectivity between the device and fixed beacons.

In this research, we used two flavors of these sensors, WeC (Figure 2) and Rene (Figure 3, left). The WeC motes have an on-board photo-sensor. The Rene motes have an expansion board; currently available expansion slots include an accelerometer and magnetometer. The Rene motes can also sense and record the signal strength of received packets.

TinyOS fits in 178 bytes of memory, propagates events in the time it take to copy 1.25 bytes of memory, context switches in the time it takes to copy 6 bytes of memory, and supports two levels of scheduling. This OS is designed for application specific devices, supporting concurrency intensive operations, and functionality modularity with minimal processing overhead. The OS targets applications that flow information through the network, acting more as routers than as workstations.

The sensors communicate with a PC through a programming board (Figure 3, right). Their OS can be installed and updated through the parallel port, and the devices can send sensor data back to the PC through the serial port.

EMPIRICAL ANALYSIS METHOD

With the objective of inferring sensor location from sensor network activity, we collected more than 15,000 sensor readings, measuring signal strength, noise, and packet loss at known distances. Each sensor reading contained ten 10-bit readings that were summed and shifted to fit in one byte of the packet sent by the motes. Using the LEDs of the motes, we were able to count how many packets the motes were sending, and comparing this to the number of packets the base station reported receiving, we were able to calculate the number of packets lost during transmission. Once we had packet data collected, we were able to determine the byte corruption as deviations from known byte values – e.g., the sensor's identification number.

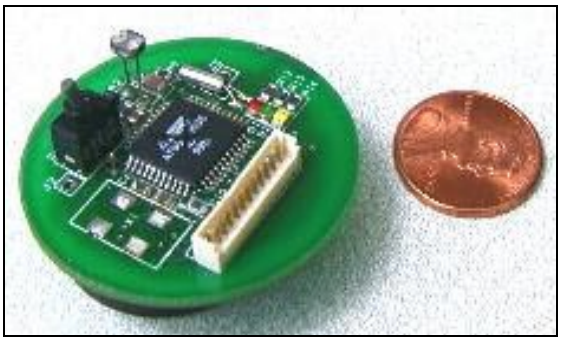

Figure 2. The WeC sensor next to a penny.

As a result of these initial tests, we discovered that there was an error in the packet packaging which returned erroneous signal strength readings. We also discovered that using a 3.3V power supply on the base station created a very noisy RFM. The Rene motes cannot tolerate 3.3 V; they seem to draw a very high current and heat up severely.

Varying the potentiometer settings of the antenna allowed us to modify, to some degree, the range of the motes. We discovered that our ideal potentiometer setting was 10kOhms. Below that, and the current was too high, and jammed the radio signals at short distances, even though it gave us a better long distance range. Changing to settings above 10kOhms had a small effect on the sensor range, shortening it slightly.

We examined the motes in a number of environments. An inherent problem in using RF signaling is the large number of environmental factors that can affect the signal. For this reason we measured the motes both in and outdoors, day and night, in clear hallways, in a typical research lab, and through a large computer cluster. We found that the Rene sensors, which have an external antenna, provided the largest range, with a maximum at 230 feet outdoors at night. The WeC motes with the internal antennas had a maximum of about 30 feet indoors. This was about the minimum range, even through a computer cluster. In these noisier environments packets were received, though many were lost and/or corrupted. WeC and Rene sensors with an external antenna had an indoor range of about 60-70 feet.

Independent Variables:

1) Mote Type: we tested both Rene and WeC motes.

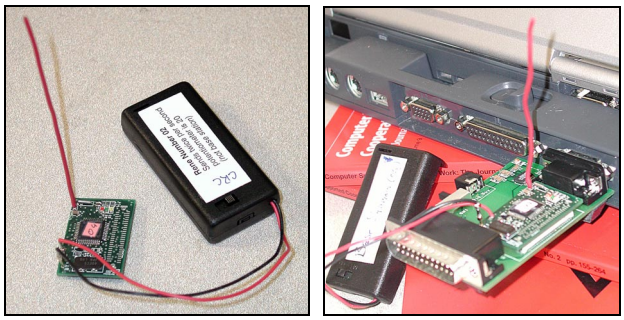

Figure 3. The Rene mote with a AA battery pack (left) and connected to the serial port of a laptop as a base station (right).

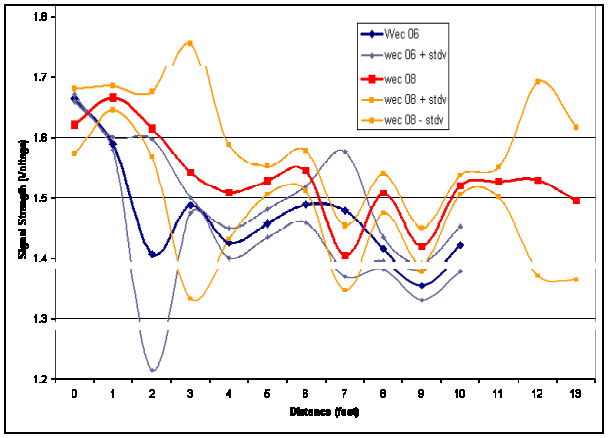

Figure 4. WeC signal strength vs. distance (short)

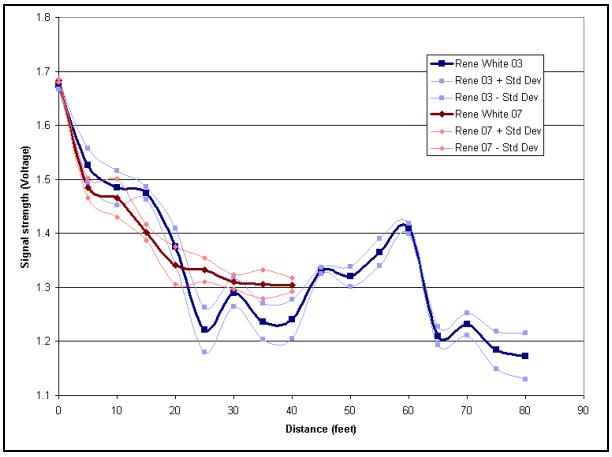

Figure 5. Rene signal strength vs. distance

2) Individual variation: We repeated many of the same experiments using different instances of the same device.
3) Environment: We tested hallway, computer cluster, research lab, outside daytime, outside nighttime.
4) Antenna type: The WeC motes are manufactured with an internal antenna that wraps around the perimeter of the device. We tested these motes with the internal antenna. We also tested both the WeC and Rene motes with an external 3.5" antenna (about ¼ wavelength).
5) Antenna potentiometer setting: The devices have a potentiometer attached to the antenna that can be controlled from software over a range of about 0-50 kOhms.
6) Base station power source: The base station can be controlled from either a power supply (with measured output of ~3.3 VDC) or from a 3V battery.
7) Distance: We tested the devices in foot-wise increments from zero feet until they lost connectivity.

Our dependent variables were:

1) Signal strength: We measured the signal strength that packets were received at by the base station. The base station calculates signal strength by summing ten 10-bit readings. This reading of 0-3Vis appended to each received packet.
2) Packet loss
3) Post-SEC-DED and CRC check byte corruption

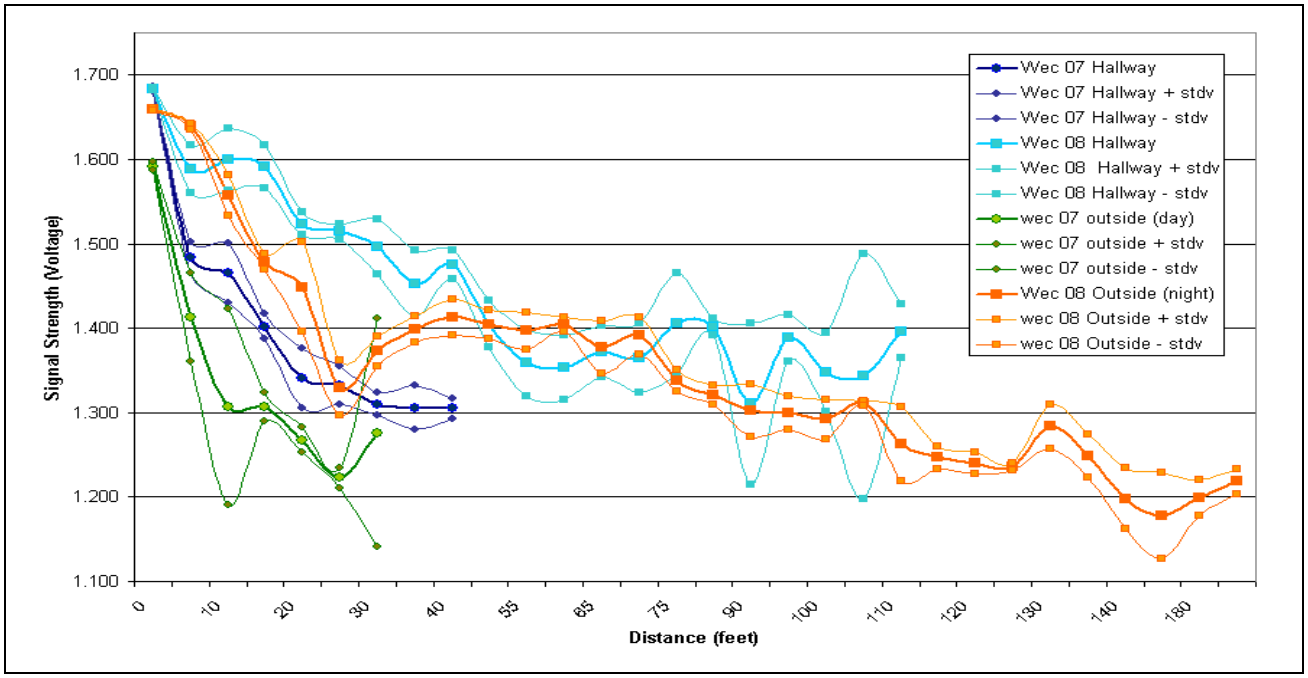

Figure 6. WeC signal strength vs. distance. Environment substantially affects the distance a device loses connectivity.

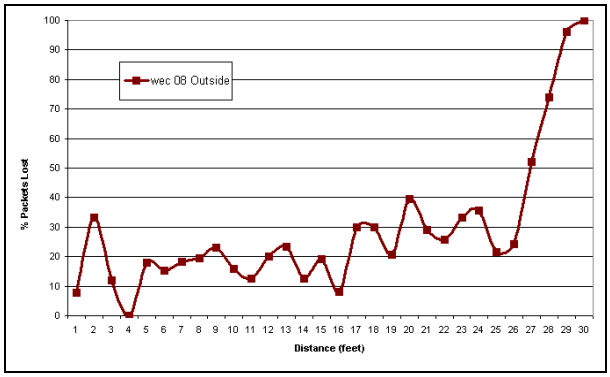

Figure 7. Counted WeC packet loss vs. distance

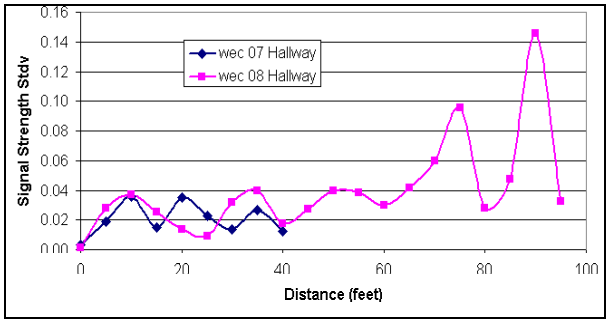

Figure 8. WeC signal strength std dev vs. distance

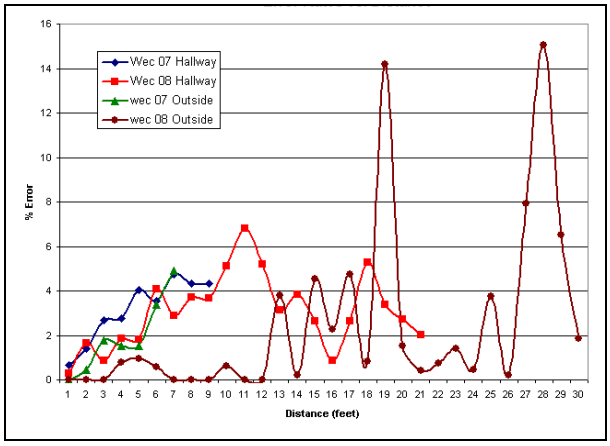

Figure 9. WeC byte corruption rate vs. distance

ANALYSIS RESULTS

In figures 4 through 9 we present some of our results of these tests. Figures 5 and 6 show that for both the WeC and Rene sensors signal strength decreased with distance in a roughly logarithmic manner. The graphs clearly indicate that there was considerable noise and byte corruption in our readings, even after error correction. As we will discuss in a later section, this noise causes considerable difficulties in creating a suitable model, even at very close distances (Figure 4). In Figure 8, a plot of the standard deviation of the signal strength vs. the distance, one can see that more than 60 feet past the base station, the standard deviation values climb towards 10% of the voltage signal strength readings.

As expected, packet loss and byte corruption did increase with distance, though not in a particularly consistent fashion (Figures 7, 9)

INFERENCE OF LOCATION GIVEN SIGNAL STRENGTH

We found that signal strength is potentially the best indicator for estimating distance between two motes. Based on our empirical measurements, we created a model for pair-wise distance as a function of signal strength using piecewise linear regression analysis. We then designed an algorithm that employs Kalman filtering [15, 16, 18] on signal strength data to estimate pair-wise distances between devices and aggregates the distance measurements with a mass spring model. A Kalman filter is a system that stores a belief about the position and velocity of an object. At each time step, it updates its estimate with the incoming data. The degree to which it incorporates a new piece of data depends upon how well the data fits the predictive model. Poorly predicted points are deemed outliers and weighted minimally. Well predicted points are weighted substantially. This is a highly successful technique that forms the basis of current successful VR trackers [11, 24].

Inferring Distance from Signal Strength

In Figure 4, we see linear and log-linear regression fits of our measurements. In this graph, the blue points are actual signal strengths received and the green lines are the represent ± standard deviation. As you can see, this curve is well approximated by a log-linear regression, shown with the red line. In this regression, we have defined signal strength (**y**) to be the following:

$$\mathbf{y} = \mathbf{C} \bullet log(\mathbf{x}) + \mathbf{v}$$

Where: $\mathbf{y}$ = signal strength

$\mathbf{X}$ = the vector [1; distance]

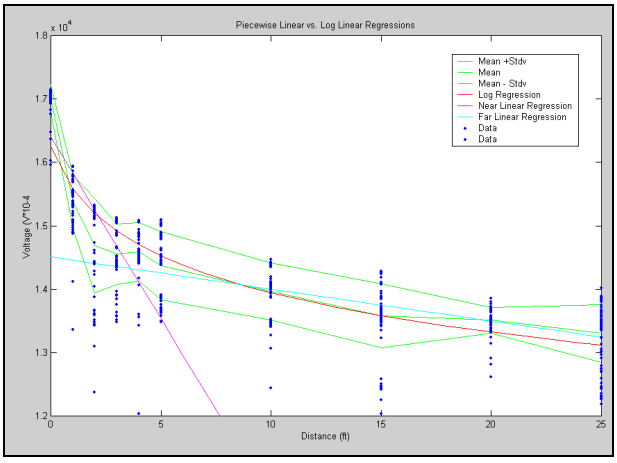

Figure 10. WeC signal strength data with both log regression and piecewise linear regression.

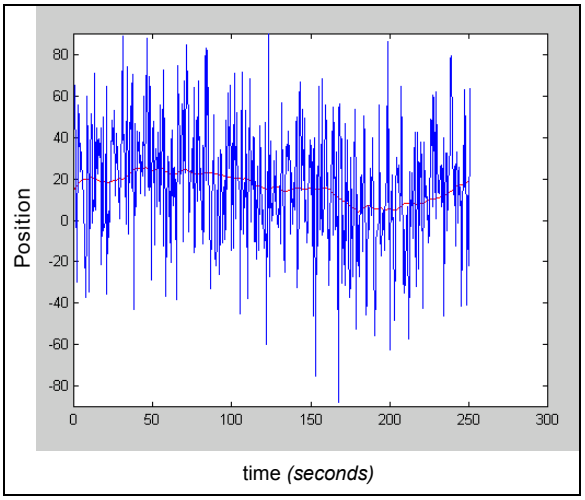

Figure 11. A graph of estimated (blue) versus actual (red) 1D position over time. The estimation uses a log regression to predict location based solely on single strength at one time step. This is clearly not useful.

$$\mathbf{v} = \text{some gaussian noise}$$

and $\mathbf{C}$ is optimized by the equation: $\mathbf{x}^t \times \mathbf{x} \times \mathbf{C} = \mathbf{x}^t \times \mathbf{y}$

We can then reverse the equation to get the probability of distance given a signal strength:

$$p(\mathbf{x} \mid \mathbf{y}) = 1/sqrt(\mathrm{s}\ \Pi\ \sigma^2) \bullet exp(\ (\tfrac{1}{2}\ \sigma^2) \bullet (\mathbf{y} - \mathbf{C}_\mathbf{x})^2)$$

Infer Distance from Previous Distance

The problem with inferring distance from only a single signal strength is that the noise in signal strength is so large that the difference between average signal strengths at two distances is almost negligible. The results of tracking an object using only signal strength are meaningless, as shown in Figure 11. The obvious solution to this problem is to take averages of signal strength over time to minimize the effect of the noise. But this does not allow us to track objects in real time.

Another solution is to filter the noise in real time, each time

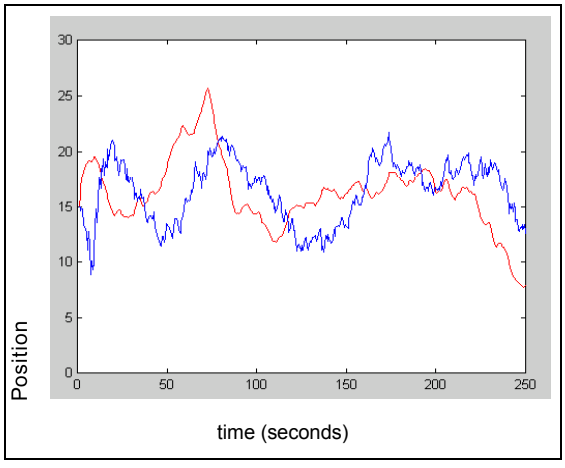

Figure 12. Kalman filter (blue) on simulated data (red). It tracks the data well, but it does introduce a time lag.

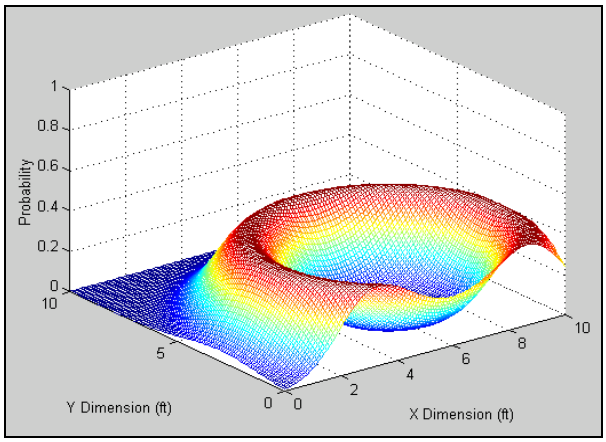

Figure 13. Posterior probability of 2D location given distance to one stationery device.

estimating our current distance and velocity, thereby allowing us to estimate our next distance and velocity. When a new signal arrives, we use it to correct our estimates. This is performed by the Kalman filter.

One problem with using a Kalman filter is that it assumes a linear, gaussian response in signal strength, and our response is log-linear and non-gaussian. To ameliorate this problem, we instead use a piecewise linear regression, shown in Figure 10 as the pink and aqua lines. These lines approximate the log regression very closely, except at the point where they intersect. Furthermore, we assume that the noise in signal strength is gaussian, even though it is not. To switch between the Kalman filters, we use the simple algorithm of choosing the larger response. This assures that we always have a continuous distance estimate.

Alternatively, we could have used a different filter, such as a particle filter, that does not require a linear gaussian response. However, the approximation performed by a particle filter may actually be equal to or greater than the approximation of a piecewise linear regression to a log regression.

Another problem with using the Kalman filter is that we need to model the movement of the object in space. To do this, we created random simulations of the mote moving back and forth, and altered the parameters until the simulation resembled the actual movement of a person in a room. We then used similar parameters in our Kalman Filter. The simulation and the results of our Kalman filter tracking it are shown in Figure 12.

Alternatively, we could have tracked a person moving in a room and empirically determined the parameters to use in the Kalman filter, but this method was sufficient for our purposes.

Infer Location from a Group of Distances

After the first two steps of processing, we arrive with a gaussian probability distribution of the distance from transmitter to receiver. If we know the location of the receiver, we can trivially transform this into a posterior of

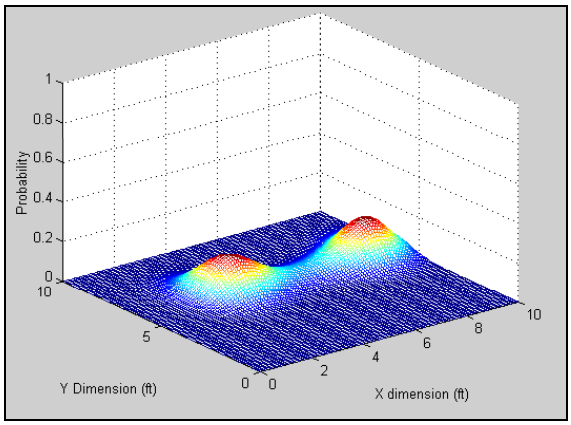

Figure 14. Posterior of location given distance for two receivers.

location given distance, *p* (location | distance), as shown in Figure 7.

If we have several receivers and know all their locations, we can multiply each posterior to arrive at *p* (location | multiple distances), as shown in Figure 6. This will always be more precise and with three or more receivers the posterior will usually have one maximum. We can trivially solve for this maximum to arrive at the expected value of location, *E* [location | a group of distances]. The posterior shown in Figure 14 is *p* (location | distance from two receivers).

Infer Location from Previous Locations

Just as before, our problem is that we have a non-linear, non-gaussian probability distribution, making the problem of tracking location very difficult. Also just as before, we have to option to make linearity and gaussian approximations or to use an approximate method, such as a particle filter.

In this case, we use a Mass Springs model, a type of approximate method. Essentially, we treat the motes as if they are connected by springs, and try to stretch or compress the springs according to our estimation of distance given signal strength.

The Next Stage of Probabilistic Inference

In practice the standard method for tracking objects is to either approximate the probability distribution with a mixture of gaussians or to use sampling methods of approximation [1-4]. However, this is usually done in systems that have a small number of sensors.

In our model, we could potentially have thousands of sensors (i.e. all other motes in the network). Some of these motes might have known locations (e.g. motes installed at a workstation) or estimated locations (e.g. other mobile motes tracking other people or objects, or simply scattered throughout the environment). This results in an incredible amount of extra uncertainty and therefore computation, mandating also an extra level of approximation.

One such approximation is to use *E* [location] for each sensor instead of *p* (location). While we did this when we used a Mass Springs model, it may be more theoretically sound to use something such as a Mean Field approximation on a Boltzmann Machine, where the values of the nodes are the **X** and **Y** coordinates, and the **Q** parameters are distances. As such, pairs of motes with small distances will tend to have the same **X** and **Y** coordinates, and those with large distances will tend separate their **X**, **Y** coordinates.

PROTOTYPE SYSTEM

Our prototype system is composed of a federation of networked sensors, a PC data server, and a PC location client. Our sensors run a slightly modified version of TinyOS that keeps track of signal strength information.

The PC server connects to the sensor network through a sensor base station that is plugged into the serial port. A Java application pulls all packets off the serial port, forwarding them over a UDP socket to the location client (which can be the same machine or a different machine).

The Java location client parses the raw sensor data. Our system uses one Kalman filter for each edge in the connectivity graph. These filters are stored in a hashtable keyed on both source and destination node (order does not matter). With each incoming packet, the appropriate Kalman filter updates the mass spring model with a new distance estimate as well as the standard deviation of that estimate. The mass spring model updates its state and is displayed with a visual user interface (Figure 15). The application allows the user to pin the location of fixed sensors, and updates the mobile sensors as new data comes in.

The Kalman filter is extended to auto-calibrate the maximum signal strength of each pair of nodes in the system. When two nodes are placed next to each other, the signal strength is highest. When the Kalman filter computes an estimate that is a negative distance (i.e. too large a strength) with a standard deviation of less than 3%, it updates its maximum strength with the new, higher measure. In practice this works quite well and begins to capture some of the variation between devices. It takes about 15 seconds to fully calibrate a device pair.

IMPLICATIONS FOR CONTEXT-AWARE SYSTEMS

The location accuracy of our system is better than infrared and passive RFID systems, which only know the connectivity graph, and improves upon wide area systems such as GPS both because it is more accurate and because it works indoors. Some of the more recent research projects, most notably Crickets [17], are substantially more accurate than our system, though they require a larger number of sensors.

Building our location infrastructure on top of the TinyOS networked sensor infrastructure has a number of positive aspects. Our system is physically much smaller than most others (with the notable exception of passive RFID tags). This system is certainly the smallest of any location system with a “general purpose” processor and operating system.

The networked sensor researchers have the goal of developing a cubic millimeter device [14]. Our system benefits from these reductions in scale to the extent that the hardware and OS offer similar functionality to the current system.

Having a processor on each device potentially allows the individual nodes to address privacy and security issues in location-aware application, currently a substantial barrier to adoption [20] [17]. Additionally, because there is no hardware distinction between fixed devices and mobile devices, a) user deployment is simplified, and b) a small ad-hoc network of devices can leave network shot, be cognizant that they are connected, and report that information when returning to the main network (e.g. three individuals head to a coffee shop for a meeting. The system could potentially know they were together).

SYSTEM IMPLICATIONS AND FUTURE WORK

Our analysis leads to several implications for the mote hardware and TinyOS. Our measurements indicated that outside of 15 feet, over 5% of the bytes received were corrupted even after SEC-DED error correction. Even 5% byte corruption is not tolerable. Packets in the system are currently 30 bytes of data and 2 bytes for error check. This rate yields an average of 1.5 bytes corrupted per packet. Error correction better than SEC-DED with a CRC check is needed. An additional 7 check bytes would enable dual-error correction. Transmitting the 7 bytes would be reasonable, however the cost of additional computation and larger decoding matrices needs investigation.

It may be advantageous to dynamically change the potentiometer settings and damping factor in the Kalman filter depending upon the distance. Lowering the potentiometer as distance increases might yield higher signal strength readings with fewer lost or corrupted packets at longer ranges. However, introducing this as another variable would complicate our filter model.

Currently a two-axis accelerometer sensor is available for the Rene motes. Early investigations indicate a) that adding acceleration input to the Kalman filter would considerably complicate our model, b) having information for only two axis limits its utility, and c) motion such as walking is barely detected. At best we could tell whether or not a person is moving, and even this information is dubious.

More precise identification of location could be done with additional hardware. It would also be worth investigating the size and power tradeoffs of adding other special purpose hardware, such as ultrasound technology similar to the Crickets and Bats, or hardware for calculating phase information about the RF signal (an additional source of information that could improve our model).

Our current mass-spring system operates with an even strength on all springs. Our Kalman filter produces both an estimated distance and an estimate standard deviation. We believe that our system would be more accurate if we set the spring strength to be inversely proportional to the product of standard deviation (embodying a form of confidence) and time since last packet received (estimates calculated from a larger number of packets are likely to be more accurate).

It also makes sense to revisit our method for measurement aggregation. One possible candidate would be Doherty's work on sensor aggregation via quadratic programming [9].

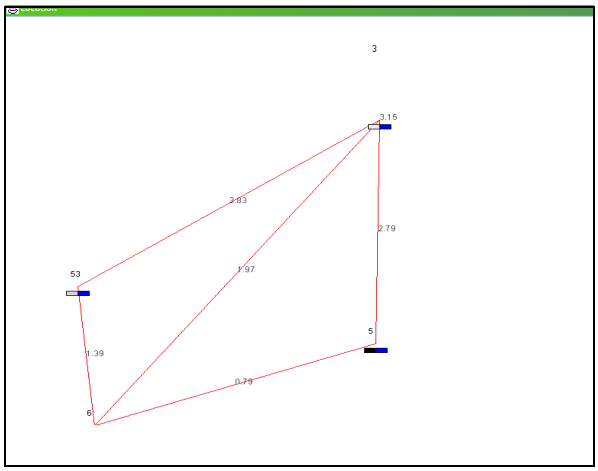

Figure 15. A screen shot of the location client with one base station (5), two fixed devices (3, 6), and one mobile device (53). Distances are along the edges, in feet.

CONCLUSIONS

We have presented empirical measurements of the RF networking capabilities of the TinyOS sensors. We used this data to build a regression model of RF signal strength as a function of distance. We designed a Kalman filter with this model, and implemented a prototype application to show the potential for a location-aware infrastructure built on top of TinyOS. We found that while signal strength varies widely across environmental conditions, a Kalman filter can account for enough of this variation to provide a useful infrastructure for context-aware applications. We believe that further research into both the devices and the filtering model could improve upon our current prototype. Our complete data and software are available for download from http://guir.berkeley.edu/location.

ACKNOWLEDGEMENTS

We'd like to give a huge shout out to Jason Hill and Rob Szewczyk for setting us up with hardware, helping us with both hardware and TinyOS difficulties, and for many illuminating conversations about our research. We'd also like to thank Mike Chen for giving us Java code that pulls data off a serial port and sends it over a socket.

REFERENCES

1. Abowd, G.D., C.G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, Cyberguide: A Mobile Context-Aware Tour Guide. *Baltzer/ACM Wireless Networks*, 1997. 3: p. 421--433.
2. ActiveRF. http://www.activerf.com
3. AEtherwire. http://www.aetherwire.com/

4. Aoki, P.M. and A. Woodruff. Improving Electronic Guidebook Interfaces Using a Task-Oriented Design Approach. In Proceedings of *3rd Conference on Designing Interactive Systems*. New York, NY: ACM Press. pp. 319-325, August 17-19 2000.
5. Bluetooth. 2000. http://www.bluetooth.com/developer/whitepaper/whitepaper.asp
6. Buonadonna, P., J. Hill, and D. Culler, *Active Message Communication for Tiny Networked Sensors*. Technical, University of California, Berkeley, CA, July 2000 2000.
7. Cooksey, D., Understanding the Global Positioning System (GPS). 2000. http://www.montana.edu/places/gps/understd.html
8. Dey, A.K. and G.D. Abowd. The Context Toolkit: Aiding the Development of Context-Aware Applications. In Proceedings of *Human Factors in Computing Systems: CHI 99*. Pittsburgh, PA: ACM Press. pp. 434-441, May 15-20 1999.
9. Doherty, L., *Algorithms for Distributed Sensor Networks*. Technical, University of California, Berkeley, CA, May 2000.
10. Federal Communications Commission, Wireless 911 rules. 2000. http://www.fcc.gov/e911/
11. Foxlin, E., M. Harrington, and G. Pfeifer. Constellation: a wide-range wireless motion-tracking system for augmented reality and virtual set applications. In Proceedings of *SIGGRAPH: 25th annual conference on Computer Graphics*. Orlando, FL. pp. 371-378, July 19-24 1998.
12. Harter, A., A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. In Proceedings of *MobiCom 1999: The Fifth Annual International Conference on Mobile Computing and Networking*. Seattle, WA: ACM Press, August 15-19 1999.
13. Hill, J., R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. In Proceedings of *ASPLOS-IX: Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*. Cambridge, MA: ACM Press. pp. 93-104, November 13 - 15 2000.
14. Kahn, J.M., R.H. Katz, and K.S.J. Pister. Mobile Networking for Smart Dust. In Proceedings of *MobiCom 1999: The Fifth Annual International Conference on Mobile Computing and Networking*. Seattle, WA: ACM Press. pp. 271-278, August 15-19 1999.
15. Kalman, R.E., A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME-Journal of Basic Engineering*, 1960: p. 35-45.
16. Maybeck, P.S., *Stochastic Models, Estimation, and Control, Volume 1*: Academic Press, Inc., 1979.
17. Priyantha, N.B., A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In Proceedings of *MobiCom 2000: The Sixth Annual International Conference on Mobile Computing and Networking*. Boston, Massachusetts: ACM Press, August 6 - 11 2000.
18. Sorenson, H.W., Least-Squares estimation: from Gauss to Kalman. *IEEE Spectrum*, 1970. 7: p. 63-68.
19. Spreitzer, M. and M. Theimer. Providing location information in a ubiquitous computing environment. In Proceedings of *Fourteenth ACM Symposium on Operating System Principles*. Asheville, NC: ACM Press, December 1993.
20. Want, R., A. Hopper, V. Falcão, and J. Gibbons, The active badge location system. *ACM Transactions on Information Systems*, 1992. 10(1): p. 91-102.
21. Want, R., *et al.*, Overview of the PARCTAB Ubiquitous Computing Experiment. *Mobile Computing*, 1995. 2(6): p. 28-43.
22. Ward, A., A. Jones, and A. Hopper, A New Location Technique for the Active Office. *IEEE Personnel Communications*, 1997. 4(5): p. 42-47.
23. Weiser, M., The computer for the 21st century. *Scientific American*, 1991. 265(3): p. 94--104.
24. Welch, G., G. Bishop, L. Vicci, S. Brumback, K. Keller, and D.n. Colucci. The HiBall Tracker: high-performance wide-area tracking for virtual and augmented environments. In Proceedings of *ACM symposium on Virtual reality software and technology*. London, United Kingdom: ACM Press. pp. 1-10, December 20-22 1999.