Fundamental Efficiencies of Mica-based Wireless Networks

Shane Erickson
Seattle University

Summer Undergraduate Program in Engineering Research at Berkeley
SUPERB, Summer 2002

Faculty Advisor: Dr. Ruzena Bajcsy
Graduate Mentor: Dan Steingart

College of Engineering
University of California, Berkeley
1. Introduction

Interest in wireless sensor networks has risen in the last few years. Their potential applications range from lighting control to emergency response, utilizing the promised attractive features of wireless communication and inexpensive implementation. The main idea behind wireless networks is to communicate environmental data among a set of nodes while acquiring useful information [1].

Under funding from DARPA, UC Berkeley has invested its resources in developing its own wireless sensor network. At the forefront of the development are Dr. Kristofer Pister and Dr. David Culler, Professors of Electrical Engineering and Computer Science at UC Berkeley respectively. One major milestone occurred on August 27, 2001 when they used 800 nodes to demonstrate a self-organizing wireless network [2]. Separate from the development of hardware and software, but equally important, is the implementation of the motes. Dr. Paul Wright of the Mechanical Engineering Department and his lab are currently developing sensor networks to monitor temperature gradients in a room to decipher comfort issues, track the location of people in a building, and even detect the movement and decipher the location of fire in a building.

The most widely used nodes developed at UC Berkeley are called Mica motes. A Mica mote consists of a printed circuit board containing a low-power 6MHz microcontroller, flash memory, and an RFM radio transceiver. Although this main hardware functions independently, a sensor board is often attached by means of the 51-pin connector. The Mica sensor board comes with a photo diode, accelerometer, microphone, thermistor, magnetometer, and buzzer. There is also a basic sensor board, which comes with no sensors to allow the user to design a specialized sensor or actuator. Figure 1 shows a Mica mote on the left and a Mica sensor board on the right. When attached, the sensor board rests on top of the mote, and the two communicate through the 51-pin connector. When operated, the Mica mote transmits its data, typically including a sensor reading, via the RFM radio transceiver at 916MHz. This signal can then be received by another mote.

The data communication between motes is quantized in terms of packets. A packet has thirty-six bytes and contains a variety of information. According to Dr. Culler, the radio transceiver can only handle 40 packets per second. To gather and move data, another mote is used called a base station. A base station is a centralized node that sends or receives data to or from other motes, and it is typically connected to a computer, feeding information through the serial port.

Figure 1. Mica mote (left) and Mica sensor board (right)
The software used to program the motes is called TinyOS. TinyOS is an event-based operating system that ‘wires’ layers of code together to create an abstract programming environment. For more information on programming in TinyOS, reference Philip Levis’ article *TinyOS: Getting Started* [3].

The flexibility of the motes and TinyOS readily gives UC Berkeley researches and others a host of possible implementations. Houses and offices of the future could potentially use wireless sensor networks to monitor light, heat, and energy consumption. This would allow more efficient methods of distributing power to be implemented. The State of California hopes to save anywhere from 7 to 8 billion dollars a year in energy costs [4], which is why Governor Gray Davis approved the proposal to establish the Center for Information Technology Research in the Interest of Society (CITRIS). CITRIS, to achieve its goals, is highly interested in the implementation of motes.

In the future, motes could also be used to monitor a building’s health during or after a disaster. If equipped, a building could give itself an instant assessment of its condition and repairs that need to be done. Rescuers could then use the building health ratings to prioritize their relief efforts. The motes might even allow rescuers to know how many people occupy a building at the time of disaster, along with their locations within the structure. [5]

In order to implement motes for wide scale use, more must be understood about their capabilities. The Mica motes and sensor boards are designed from off-the-shelf parts, each part with its own specified limitations. Each weak point could be investigated separately and an analysis of its overall effect analyzed. This information, however, is not immediately useful to the engineers implementing mote networks. Instead, immediate research should focus on researching the overall limitations of the Mica motes in a wireless network, providing quantitative results useful to engineers. Since the motes communicate through radio frequencies, their overall ability can be determined by the accuracy of sending and receiving packets.

In the interest of determining the characteristics of the motes when the quantities and transmitting rates vary, this paper investigates the efficiency of each situation when using 1 to 5 motes programmed to transmit at rates up to 128 packets per second.

2. Experimental Methodology

2.1 Experimental Setup Overview

A group of 1 to 5 motes transmits packets at speeds from 1 to 128 packets per second with every combination tested therein. Another mote attached to the serial port via a programming board is programmed to receive packets and forward them to the PC. The Serial Forwarder, as part of the TinyOS package, opens the serial port and allows access to the incoming packets. A Visual Basic script is used to interpret the packets, and any vital information is output to a file. From the file, specific data is extracted and analyzed in Excel. Figure 2 gives a block description of the setup.
2.2 Experimental Setup Details

The idea behind running efficiency tests on the Mica motes is to see how they will behave in actual conditions. To achieve this, a modified version of the program sens_to_rfm included in the TinyOS package will be used. Using an already written program is justifiable since most users of the motes are not software programmers and will need to heavily depend on prewritten software. The sens_to_rfm program is used because it is straightforward; the program simply obtains a light sensor reading from the photo resistor in eight-bit resolution and transmits that value in a packet through its transceiver. The program is also collision resistant. If one mote detects another mote sending a packet when it wants to, the original mote waits and sends its packet later.

Two minor modifications are made to the sens_to_rfm program. First, in the file INT_TO_RFM.c, a counter variable is installed so every time a packet is sent, the count variable is incremented by one integer. This integer value is put into the transmitted packet so the receiving end can identify how many packets are sent from a particular mote. The other modification allows the red LED to turn either on or off, whatever the opposite of its current condition is, whenever a packet is sent from a mote. To make this work properly, minor additions to the code in the INT_TO_RFM.desc file are necessary to link or ‘wire’ the appropriate components. See Levis’ TinyOS paper for more on linking components [3]. The rate at which the mote can send data is set in the SENS_OUTPUT.c file, and the rates used are the powers of 2 (i.e. 1, 2, 4, 8, 16, ...), and the units are packets per second. Also in this file, 8-bit photo resistor resolution is specified from a range of 1 to 10 possible bits.

The base mote has the generic_base_high_speed program installed to receive packets. This program is also included in the TinyOS package. The base is attached to the programming board, and the programming board is connected to the serial port of a PC. This allows any incoming packet to the base to be forwarded to the PC via the serial port. The Java program Serial Forwarder, included with TinyOS, is used to open the serial port, and Visual Basic scripts organize the incoming packets. Important data is put out to a file. The data from the files can be extracted into Excel for analysis.
2.3 Description of Experiments

The first test performed involves determining the relationship between how many packets are programmed to be sent from the mote per second and the efficiency. This is done individually with 1 to 5 motes. Efficiency is defined as the number of useable packets received by the base divided by the total number of sent packets. To begin, five motes are programmed with the modified sens_to_rfm_program to send at a rate of one packet per second. No sensor board is attached to the mote since sensor readings are not necessary to calculate efficiency. Following, the mote is placed next to the base well within reception range to receive the sent packets. Visual Basic is used to inspect the received packets and determine the overall efficiency, and it stops collecting packets after 100 have been sent. The details of Visual Basic are discussed in the next section. This process is repeated ten times and the results averaged to obtain an overall efficiency.

After the tests for one mote are complete, two motes are used, and they are placed equidistant to the base. The entire process of obtaining the efficiency is repeated. The total number of packets needed to stop the testing in Visual Basic increases to 200. After completing the tests for three, four, and five motes, they are programmed to send at two packets per second, again starting at one mote and working up to five. The speeds continue increasing by powers of two until 128 is reached, and the total number of packets needed to finish a test increase to allow more time to collect data and therefore being more accurate. Ideally, the tests run for infinite time, yielding a true efficiency, but since this is not practical, averaging multiple runs is sufficient.

As a note, whenever one mote is used, its mote ID will be set to 1. When two motes are used, their ID’s are 1 and 2 and so forth. This allows the PC to know exactly how many packets are being received and from which mote each one comes from.

The second test compares the actual packets received per second by the PC and the programmed packets per second of a mote. If there are any discrepancies in the first test, this one may help to determine a possible problem or cause. Since only Visual Basic is needed to determine the relationship, details will be discussed in the next section.

2.4 Visual Basic and Excel Methods

Visual Basic is the experimenter’s choice for determining the efficiencies needed for the first experiment. To begin, imagine the Visual Basic program as a black box; it takes in a packet, determines if it is valid, and if it is, deciphers which mote it comes from. A valid packet is one that does not have any errors, or in other words, has perfect transmission. Visual Basic can accurately determine this because it knows what most bytes in a packet are supposed to be. Recalling that there are thirty-six bytes in a packet, in the modified sens_to_rfm program the first four are the header, the fifth is the light sensor reading, the sixth is the mote ID, the seventh is initialized to zero, the eighth and ninth contain the counter variable that specifies how many packets have been sent, the next twenty-five contain what is usually data but in our case initialized to zero, and the last two contain error checking values used in transmission known as CRC. Of all thirty-six bytes, thirty are known values that are used to filter incoming packets to
determine their validity. The known values are the four-byte header, byte seven, and the twenty-five data values initialized to zero. The filtering is done through a byte-by-byte comparison, and if there is a discrepancy between the expected and received values, that packet is classified as ‘junk.’ If the mote ID byte is larger than the total number of motes, that packet is also labeled as junk. The two error checking bytes are not used in determining a packet’s validity and neither is the photo sensor byte since it constantly changes despite no sensor board being used during the experiments.

There are six major categories calculated by Visual Basic:

- **Total Packets Sent by the Motes**: This is calculated by adding the bytes that contain the count variable from each individual mote together.
- **Total Packets Received by the PC**: This is calculated by counting the number of received packets.
- **Total Valid Packets Received by the PC**: This is calculated by counting the number of received packets that do not contain any errors as specified above.
- **Total Junk Packets Received by the PC**: We obtain this value by incrementing it by one integer each time a packet is not valid.
- **Total Packets Not Received by the PC**: From subtracting the total packets received by the PC from the total packets sent by the motes, this value is determined.
- **Efficiency**: The ratio of total valid packets received by the PC to the total packets sent by the motes yields this value.

The above six values, in addition to each individual mote’s number of received as valid packets and total sent packets, are output to a file after each run. Once a run is complete, the efficiency value is immediately input into Excel. By doing this, we instantly determine the standard deviation for each type of test. If the standard deviation becomes large, we perform more trials to make sure we have a representative distribution. Another method is to increase the total number of packets needed in Visual Basic to end the test. As a note, all graphs generated in Excel will have a logarithmic scale for the x-axis to make them easier to understand and interpret.

For the second experiment, Visual Basic calculates the packets received per second for only one mote. This is done by simply counting. After one second has elapsed, Visual Basic outputs how many packets it receives and resets itself to zero for the next second. This process continues until interrupted by the user. A graph of received packets per second versus programmed packets sent per second is easily achieved in Excel.

3. Results

3.1 Efficiency vs. Programmed Packets Per Second

As previously discussed, a series of tests are run for each speed from 1 to 128 programmed packets per second (abbreviated 128pps from here on out) for 1 to 5 motes. This produces forty
Table 1. Number of packets and trials required for each of the forty efficiency tests

different types of tests. Initial testing is done where each test is performed three times (ten times at some of the faster transmitting rates), and the results averaged. During the official testing, if the standard deviation increases dramatically, more trials are performed and/or more packets are required to end the test to produce a more representative distribution. Table 1 explicitly states the number of packets required to be sent and the number of trials performed and averaged for each test.

The forty individual tests produce the graph shown in Figure 3. Efficiency graphs with error bars, as determined by the standard deviation, for all number of motes are found in Appendix A.
Figure 3 reveals several interesting characteristics. For 1 mote, the efficiency stays relatively close to 100%. For 2 motes, the efficiency is virtually 100% for the lower sending rates and mildly falls off at the higher ones. The trend lines for 3, 4, and 5 motes tend to follow the same pattern. Between 1 and 4pps they remain relatively constant producing nearly perfect efficiency. As the trend lines approach 8pps, they tend to decrease. Interestingly, at 32pps the trend lines rebound and never reach the low point again.

For the tests involving one mote, the standard deviation remained small and constant. With 2 motes, the standard deviation increases significantly at 16pps and never decreases significantly after that. Again, we see correlations among the 3, 4, and 5 mote tests. The standard deviations dramatically increase as the efficiencies decrease at 8 and 16pps. During the rebound at 32pps the standard deviations decrease. After 32pps, the respective standard deviations remain large.

Since the variances between 8 and 64pps are large for 3, 4, and 5 motes, the results need to be verified to make sure they are repeatable. All the efficiencies for rates slower than 8pps are virtually 100% so those values are not verified. 128pps is not verified because their standard deviations tend to be small. A new set of motes is selected and the tests are run again with the same conditions as specified in Table 1. The only exception is that 10 trials are performed for each type of test. The results of the verification process can be found in Appendix B, and they demonstrate the repeatability of the experiments. The only major discrepancy between the official and verification tests occurs with 3 motes running at 64pps. To further verify this data value, another set of randomly chosen motes is selected, and the trials for 3 motes at 64pps are repeated. A solid box found in Appendix B Figure 10 denotes this value.

Interestingly enough, all the verification results virtually mirror the official testing with only a minor vertical axis (or efficiency) offset. The initial tests are also included in the figures of Appendix B to show repeatability. Strong emphasis should not be placed on initial trials because they are determined by only averaging three values together (ten at some of the faster sending rates) and are therefore not as accurate as the official and verification trials.

3.2 Actually Received vs. Programmed Packets

This experiment demonstrates how many packets are being received by the PC versus how many the mote is programmed to send. Figure 4 is a graph of these characteristics.

We see the linear relationship between both axes up to 16pps. Then the graph flattens; the programmed rate increases but the rate at which the PC receives the packets remains constant. After 32pps, the rate at which the mote is programmed to send packets increases much more rapidly than the number of packets received by the PC. These results are confirmed using another TinyOS program called Chirp.
4. Discussion of Results

Since the radio transceiver can only handle 40 packets per second, the efficiencies in Figure 4 should predictably decrease. For example, with 1 mote at 64pps we would expect the efficiency to be 62.5%, but instead it is 98.0%. Even at 128pps with 1 mote, the efficiency is 97.5%.

Even though the graph in Figure 4 depicts the number of packets received per second by the PC on the y-axis, we can use it as an estimate for the number of packets sent per second by the motes. This estimation works because Figure 3 demonstrates that for all single mote transmitting rates, the efficiency is nearly 100%. Looking at Figure 4 for 128pps, the mote is only sending roughly 26 packets per second. This explains the 97.5% received/sent efficiency. It becomes clear that a new term for efficiency, actual/programmed should be considered in addition to received/sent in any future experiments.

Although we expect a falloff in the trends as they reach the threshold of the radio on the base, for 3, 4, and 5 motes they occur earlier than expected. For 3 motes at 8pps we expect a total of 24 packets to be transmitted, well within the threshold range. Also, with 5 motes at 8pps we expect a total of 40 transmitted packets, just barely within the threshold range. The discrepancy is that the efficiencies are 96.2% and 71.8% for 3 and 5 motes respectively when they should both be just under 100%. This is likely due to interference of radio signals between the motes and
possibly other outside sources (an old generation 900MHz WLAN operates in neighboring Soda Hall, and there is a proliferation of 900MHz phones through Etcheverry Hall).

As visually observed, if two motes are sending out packets at virtually the same rate, the base rarely receives one of the mote’s signals. This partially explains why we have lower efficiencies than expected, but it also demonstrates weakness in the software and/or hardware. As we understand it, the modified sens_to_rfm program does not allow a mote to send out a packet if another one is currently being transmitted. This program does not work flawlessly, then, if a mote is sending out packets not received by the base when the base has not yet reached its threshold limit of 40 packets per second. Even if the motes are slowing down, the base should receive most of the data.

There are also instances of efficiencies being greater than expected. Using Figure 4, for 4 motes at 64pps we anticipate a total of 88 packets sent per second. Knowing the limitation of the base radio of 40 packets per second, we estimate the efficiency to be 45.5%. The measured value is actually 74.5%. There are a number of these examples, and this implies that the already reduced transmitting rates for values like 64 and 128pps are attenuated even further when more than one mote is present. Perhaps the time spent idle, occurring when a mote waits to send its packet when another mote is presently transmitting one, is amplified at the larger programmed sending rates. Another possibility of the motes slowing down is reaching the maximum processing power of the Atmel microcontroller chip.

The increase in efficiency for 3, 4, and 5 motes between 16 and 32pps is a curious phenomenon. From Figure 4 we know that for a single mote, 16 packets per second are transmitted at both of these transmitting rates. The efficiencies, then, should be the same. Interesting correlations may exist between these speeds and how many motes can be used, but more thorough investigations and experiments need to be performed to identify and quantify them.

## 5. Conclusions

Using a wireless network of Mica motes for fast-changing conditions where transmission rates on the order of 16pps or higher are needed is not suggested; missed or unusable packets are inevitable. Because of this, use motes in applications that are not sampling rate sensitive, for example, temperature or standard lighting. If fast transmission speeds are desirable, increasing the number of base motes and placing them in strategic locations should increase efficiency.

Efficiency may also be increased through asynchronous broadcasting. This is particularly important when using the motes in emergency situations. For instance, running a mote inside a smoke detector would require sending packets only if smoke is detected. The send-if-only-necessary method would limit the data a base would have to process in any application; in emergency situations it is of utmost importance to maintain a clear channel to the base.

Using different motes may also yield different efficiencies. As demonstrated in Appendix B, the experiments performed are repeatable but the official and verification testing results were vertically shifted from each other. Since the two tests were done with different sets of motes,
efficiencies are dependent on which motes are used. This should be taken into consideration when using them.

6. Future Work

Two types of research should be performed to draw further conclusions. The first involves black box tests. To better understand the motes we need to know how they act overall regardless of any individual limitations. In the case of this research, the radio on the motes may only be able to handle 40 packets per second, but other hardware or software issues may decrease that amount. Specific tests should include but are not limited to determining how many packets per second are received by the PC as well as sent by the motes for varying quantities of them, how more than 5 motes act while using a single base, and how programming the motes to transmit at different rates affect each other. The second type of research entails describing what is actually happening to cause such unintuitive behavior at the software and hardware level. Perhaps the microcontroller is too slow or the software is not optimized. These tests, however, are not as immediately useful to the engineers who implement mote networks.

7. Acknowledgements

I would like to thank the following:

- Dr. Ruzena Bajcsy, my faculty advisor, for imparting her wisdom and experience to me, carrying over into my work. The time she provided me was invaluable.
- Dan Steingart, my graduate student mentor, for his time, support, and guidance in this research experience. Without him my direction would have been lost.
- Dr. Paul Wright and his students for allowing me to work in their lab, in addition to their humor and companionship.
- The other SUPERB students for being there, dealing with me, and sharing the experience of eight weeks in Berkeley.

8. References

Appendix A

This appendix contains the individual mote efficiencies for 1 to 5 motes with error bars included. The error bars indicate one standard deviation in either direction of the mean. In the case that a standard deviation caused the error bar to be greater than 100%, it was cut off since efficiencies cannot be greater than 100%.

Figure 5. Overall efficiency for 1 mote with a range of 1 to 128pps
Figure 6. Overall efficiency for 2 motes with a range of 1 to 128 pps

Figure 7. Overall efficiency for 3 motes with a range of 1 to 128 pps
Figure 8. Overall efficiency for 4 motes with a range of 1 to 128pps

Figure 9. Overall efficiency for 5 motes with a range of 1 to 128pps
Appendix B

This appendix contains the mote efficiency verifications for the 3, 4, and 5 mote tests. The 3 mote verification test involves an inconsistency for 64pps between the official and verification test. Therefore, a second verification of that value is done. It is indicated in Figure 10 by the solid box.

![3 Mote Efficiency Verification With Inconsistent Value Confirmation](image)

Figure 10. 3 mote efficiency verification showing the initial, official, and verification test data. The solid box indicates the inconsistent value confirmation.
Figure 11. 4 mote efficiency verification showing the initial, official, and verification test data.

Figure 12. 5 mote efficiency verification showing the initial, official, and verification test data.