

Reliability of Mica2 Motes Using Generic Communcation

Mark Daniel Krasniewski
Dependable Computing Systems Lab
School of Electrical and Computer Engineering
Purdue University.

To test the design of the Deluge protocol and set up a pretest to determine how many messages Deluge or any other type of protocol sending out downloadable would have to send, I constructed a simple 2 hop network using the 3 mica2 motes I had. The objective of this network is to see how reliable these motes were less than 36 feet away from each other. This distance was chosen because I was under space restrictions with LOS within my work area, and a short experiment in an open space at 40 feet showed difficulty getting a consistent signal. So I set up a controlled environment with close-knit motes.

The first order of business is the design of this simple reliability test protocol. Necessarily, motes would like to send some data over the wireless, so to create a method of checking that messages are conveyed reliably we need a way to compare data at the source and the sender. Moreover, we want data that can fluctuate, hopefully crossing the spectrum of possible outputs. Therefore, I incorporated the mica sensor board and photo sensor, so that in taking readings the motes would have a source of relatively “random” data. I found that a TV next to the photo sensor with some arbitrary light switch flips can be surprisingly effective at generating dramatic changes in brightness that appear somewhat random.

Next, I need to verify that the correct data is transmitted over the network. It is possible to run checksum and more complicated procedures on a base station node to verify the data transmitted over the network. However, it is more useful if I can see the entire range of what is transmitted over the network and the motes aren't conducive to that, so running a java program called Listen can snoop on the network (via a base station mote) and tell me everything that it hears. So I used this program to snoop on all radio communications and in this way collected every packet sent and could verify that the correct information was sent. I then set up two motes in a network with this third snooper. The two network motes sent messages among themselves and then to the snooper. One mote in the network collected photo readings and forwarded those to the other network mote, and then that mote sent the reading to the snooper. Due to the small size of the network, the middle hop wasn't entirely necessary, but it did create an opportunity to verify multihop networking and to gather an extra data point in node-to-node transmission.

For the packets the network sent, there are 5 different bytes for header information and then a predetermined number of extra bytes. I set up packets with the following information:

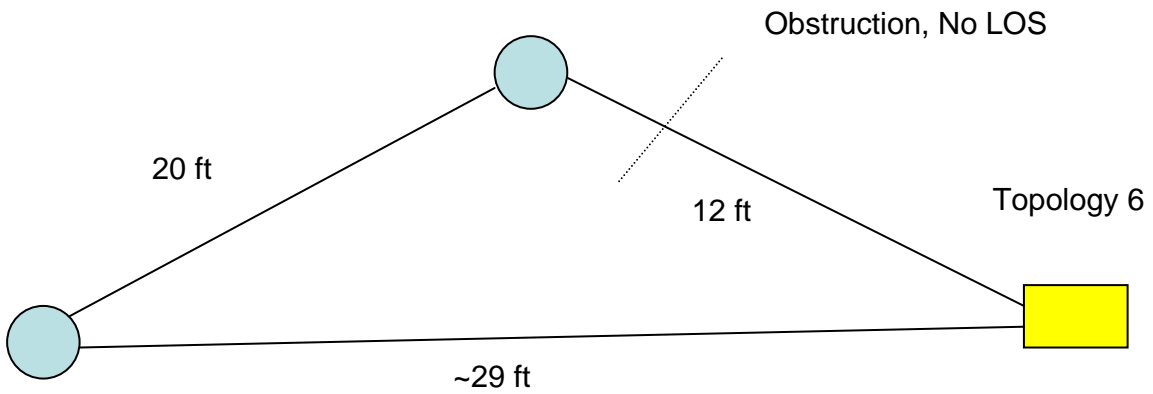
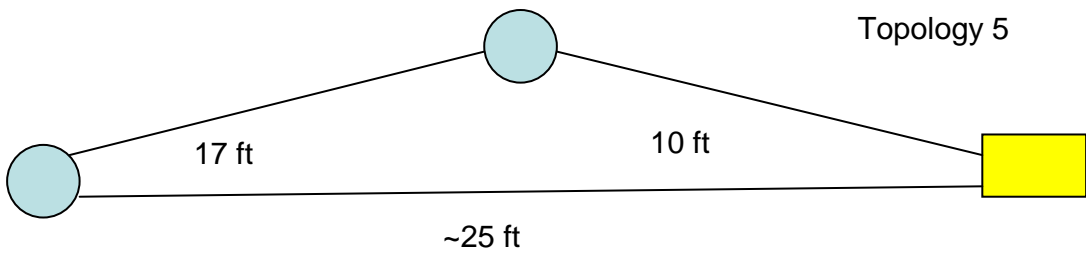
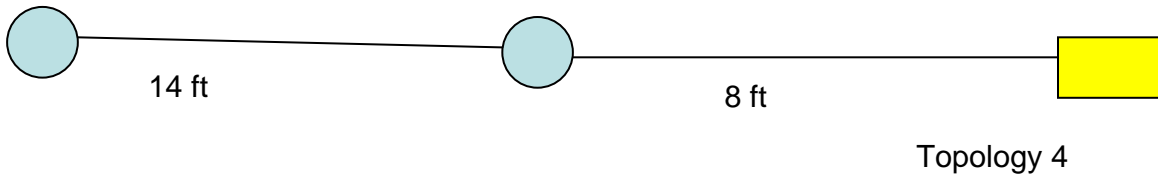
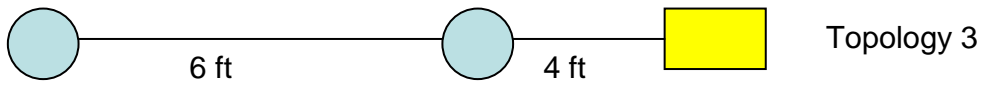
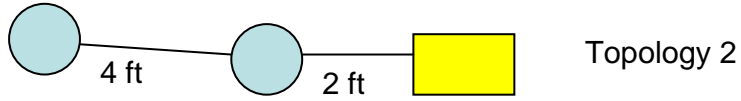
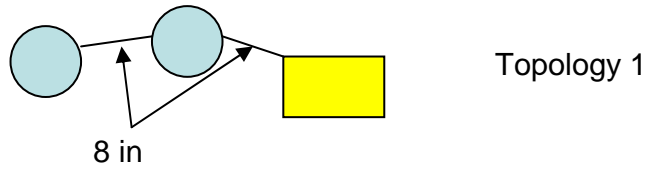
Destination Address (2 bytes, header info), handler ID (1 byte/useless, header), message length (number of extra bytes – 1 byte, header), Group ID (1 byte, relatively useless, header), Source mote (2 bytes, useful) Data Reading (2 bytes, useful to verify that the

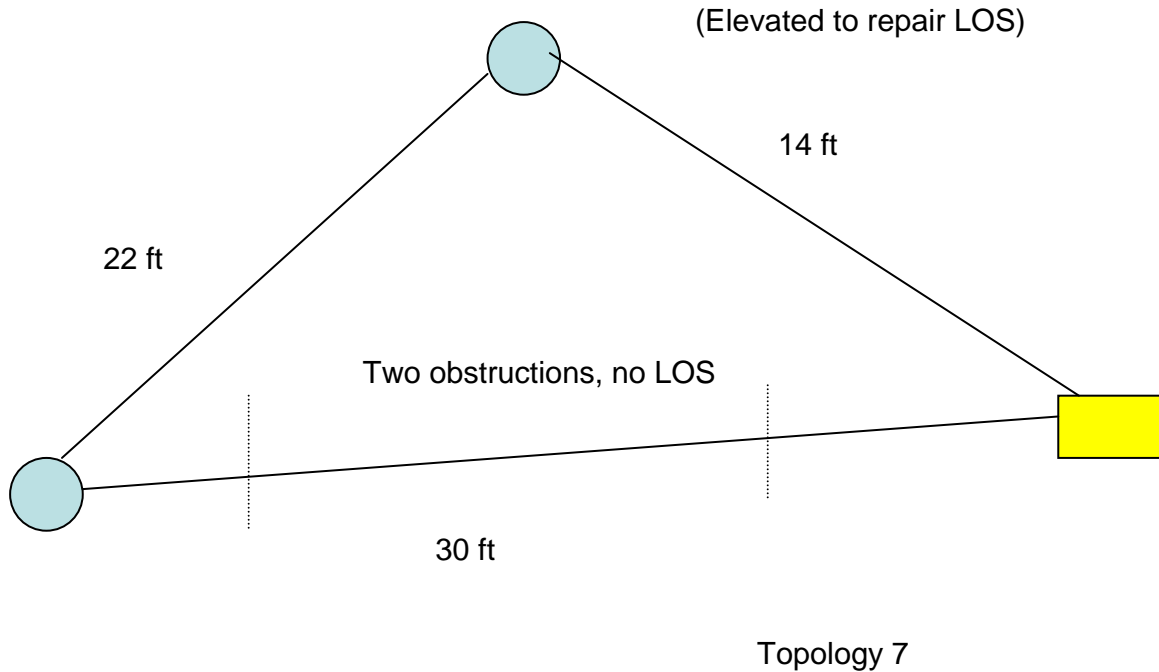
same data is sent by both motes), message number (2 bytes, to check on how many messages a mote has sent to the base station or other mote), incoming message number (2 bytes, used by the middle mote to see how many messages the far mote thinks it has sent to the middle mote, and based on discontinuities in this count, I can see how many messages the middle mote has actually received).

Therefore, each message was a mere twelve bytes long. Packet size is determined to be 36 bytes by the software, but it wasn't necessary to send this complete size of packet. I am not certain yet whether longer packets would impact reliability; this is open to both experimentation as well as examination of the underlying software. It is possible that the motes send 36 bytes every time and drop extra bytes, as 36 is the constant set in the radio software files. I need to examine this further.

After designing the experimental method, I determined how to evaluate the reliability of the motes. I would send photo-sensor readings over the wireless network and use Listen to see how many of those messages reached the base node without losing quality in the data. So I set up the sensing mote to take light readings once every 1.5 seconds. Immediately after a reading the mote would send the reading value (2 bytes, 16 bit integer) to the middle mote. Incidentally, this reading would usually make it all the way to the base node. Furthermore, each sensor reading was tagged with a 16 bit message number, starting at 1. Once the middle mote received the sensor reading, it immediately forwarded this message to the base station. It forwarded the reading and the message number from the sensing node. It also forwarded its own message number, indicating how many times it had forwarded a message from the sensing mote. Ideally, these two numbers would remain equal throughout the sampling period. After running an experiment for a certain time – usually a few minutes or more – I then turned off the motes and placed them in different spots. All in all, I used seven different topologies for this network, increasing distance between motes and possibly inserting obstructions.

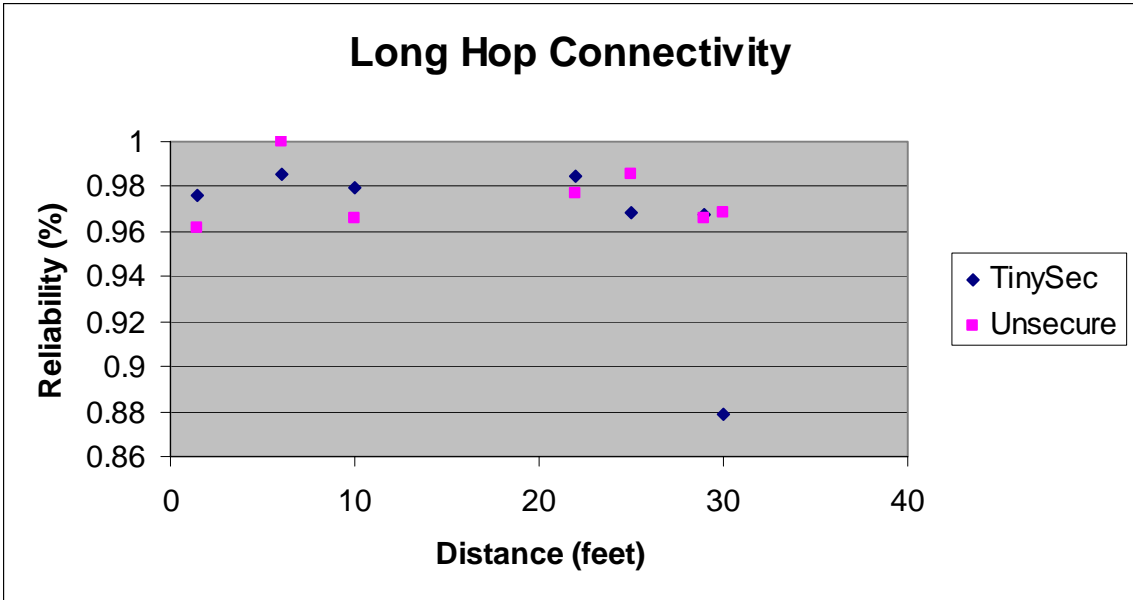
Here is a brief illustration of each topology, blue is a network mote, yellow base station:



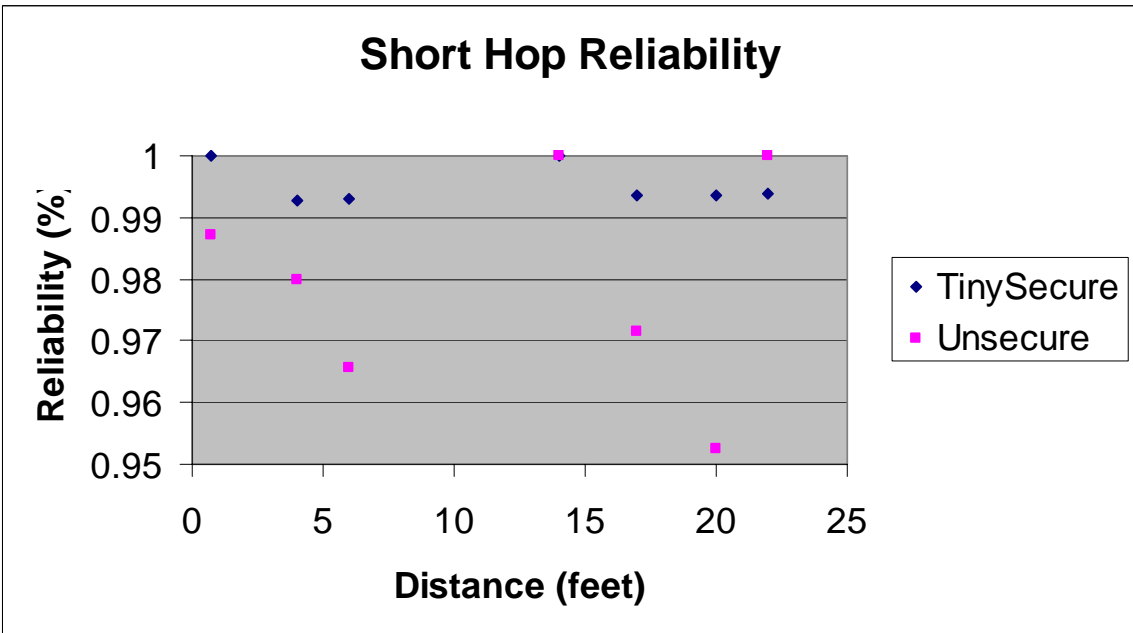


Given these topologies, we get some understanding of distance to the reliability of the motes. Naturally, there is a much larger range of possibilities, but these topologies incorporate reasonable range and incorporate some type of obstruction to make things a little interesting (obstructions were typically sofas or dry wall).

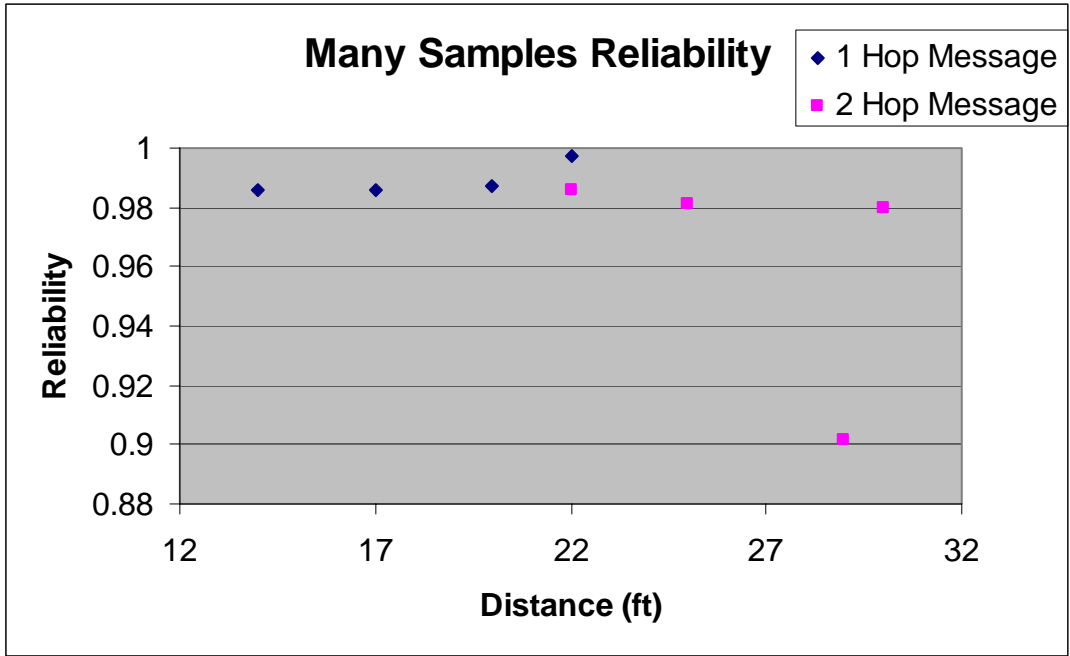
With this topologies constructed, I decided to take samples, but with two different types of networks. The newest CVS snapshot of TinyOS allows the user to easily implement TinySec, the secure mote communication protocol. The primary influences of this protocol are energy consumption, packet overhead, and security, but it may be possible that these constraints would affect the reliability. The packet overhead is 4 bytes to transmit the MAC and key for these experiments. So after setting up these two different test protocols, I began running experiments. Some basic graphs follow along with some explanation. These graphs typically involve a few thousand samples to generate one final reliability figure. Many more experiments would be needed to generate a nice range of patterns and this work remains to be completed.



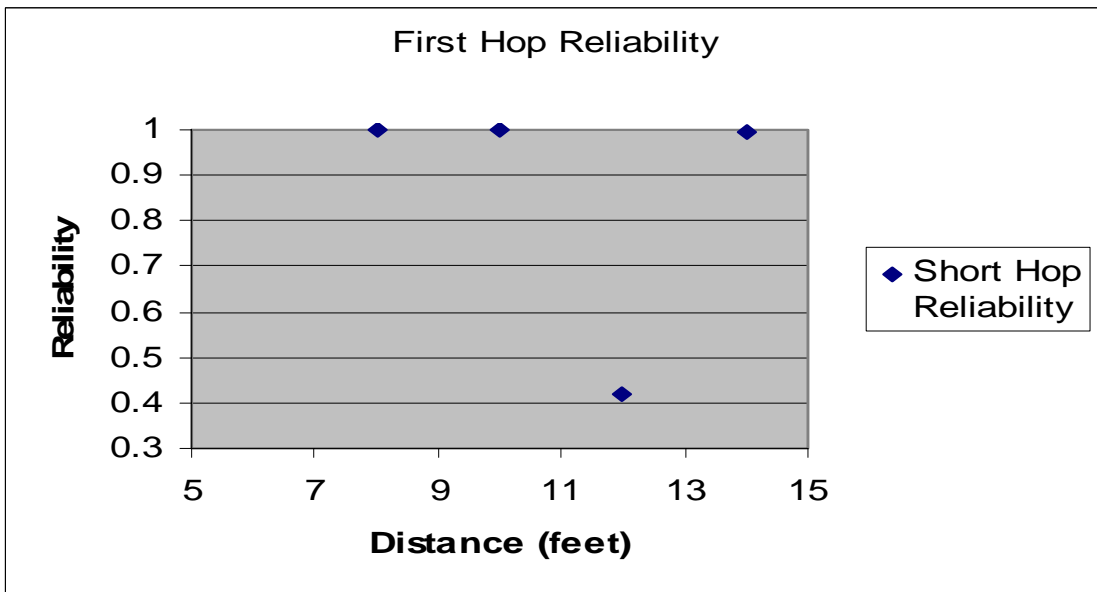
This graph demonstrates the reliability of the two different protocols over the “long hop,” the hop from the sensing node to the base station. The topologies can be seen in that each topology from 1-7 has a slightly larger distance, so the distribution of the data points along the graph also indicates topologies 1-7. Each data point is the number of messages seen by the base station divided by the number of messages sent by the sensing node. In no case did the data received by the base station differ from that received by the middle mote. Each data point is 200 samples.



The graph above demonstrates the “short hop,” or the hop from the sensing node to the middle node. Again, each data point is 200 samples, and it appears that TinSec may be slightly more consistent in its behavior.



The above graph demonstrates the reliability of a greater number of samples (2000 each) for topologies 4-7. The blue represents the sensing to middle node hop, the pink the “long hop.” Again, topologies are indicated by increasing distance. Here we see a relatively stable pattern of reliability in the “short hop,” and mostly stable although slightly decreasing stability in the “long hop.” The outlier is an interesting case we’ll discuss in the next graph.



The large number of samples in this second set of experiments allowed up to take some more readings on the reliability of the middle mote to base station reliability. Typically, we had had 1-2 errors in the first set of examples, indicating 99.5% accuracy, and I chose not to include those results. In this set, however, we had 99% reliability for all but topology 6. I suspect that this occurs due to the obstruction present in topology 6, as almost all missed packets fall consecutively. This demonstrates the extreme impact line of sight can have on the reliability of these motes. Otherwise, this graph demonstrates nearly 100% reliability at a range of 8-14 feet.

Brief tie-up: I'd estimate that at less than 12 yards the motes are ~98% reliable. Given the limited number of topologies used, it's difficult to make more useful conclusions than that. It's possible with a few thousand more samples at intervals of distance more closely approximated we could get a better picture, and this could be the next experiment. More importantly, we could see in this experiment the possibility of creating a download program with some reliability but perhaps a more energy conservative technique than Deluge, as we see here that the motes maintain a relatively high level of reliability. Here it has also been demonstrated the effectiveness of multihop communication, even if on a small scale. I think it may be possible to design a protocol that picks nodes to disseminate data over a network and limit the number of messages sent (instead of exhaustively querying the entire network). Given a small scale network, or at least one that can guarantee relatively close hops, messages can be conveyed fairly reliably. The current data transfer incarnation uses many messages among all nodes.

Moreover, given the larger packets of TinySec, I don't think that packet size has a noticeable effect on reliability (to what I can tell so far). It also might be interesting to explore just how reliable nodes are at 0-15 feet. If we can consistently get 99%+ reliability, they may make a mean network with little need for many extra messages.