

CS 240E Assignment 2: Communication

Due: 11:59 PM, February 20, 2007

1 Basics

This assignment has two purposes. The first is to introduce you to datagram based communication and the challenges of low-power wireless. The second is to give you experience in defining an experimental methodology, running the experiments, and writing up the results for others to read. The assignment correspondingly has two parts.

This assignment calls for a group of two to work together. There are two different communication approaches which each group has to implement, evaluate and compare. The hand-in for this assignment is a 4 page document describing your results.

2 The Problem

This past year at SenSys, there were two papers on techniques for improving the energy efficiency of communication. They aren't ground-breaking papers that will fundamentally change the field, but they both use a simple idea and apply it well. Since the ideas are simple, they're feasible for you to implement in the timeframe of this project. Each person in the group should implement one of the schemes. Read the paper to understand the approach. You can download the papers from the assignment page on the cs240 website.

2.1 Slicing and Dicing

The first paper, entitled "Datalink Streaming in Wireless Sensor Networks," argues that the problem with long packets is a single bit error can make you lose all of the data. As a packet grows, the probability that no bit error occurs asymptotically approaches zero. For example, imagine that the probability of any bit having an unrecoverable error is e . Then the probability that a packet of length n has no such errors is $(1 - e)^n$. If there's a bit error in the packet, then the packet checksum will not match up, the radio stack will assume the packet is irrecoverably corrupted, and drop it.

In reality, bit errors are a bit more complex than this. The CC2420, for example, actually sends 8 bits for every data bit, such that it can recover from a reasonable number of errors. In effect, that just reduces e .

Bit errors suggest that you send short packets. But sending a packet has overhead: media access backoff, headers, etc. So from an energy efficiency standpoint, you want to send long packets. There's an inherent tradeoff here.

The approach the first paper proposes is to send long packets, but to embed multiple checksums in the packet. Normally a packet looks something like this:

Header	Data	Checksum
--------	------	----------

Where the checksum is computed over the header and entire data payload. In this proposed approach, instead you do this:

Header	HC	Data1	DC1	Data2	DC2	Checksum
--------	----	-------	-----	-------	-----	----------

Where HC is "header checksum" and DC is "data checksum." Note that the paper doesn't actually have a header checksum: this is totally crazy and badly thought out on their part. But that's a least publishable unit for you. The idea is that if bit errors corrupt Data1 or DC1, then only that part of the packet has to be discarded.

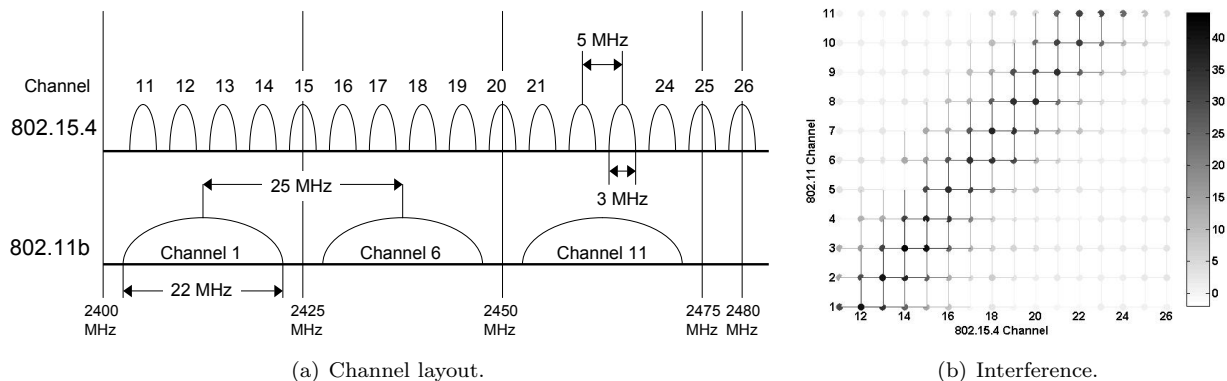


Figure 1: Channel-by-channel interference strength of 802.11b with 802.15.4. The values in the interference plot represent the difference between ambient RF signal strength observed by an 802.15.4 node when the network is quiet and the signal strength observed during a transmission by an 802.11 node that is nearby.

2.2 Compression

The second paper, entitled “Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks ,” explored different compression algorithms for sensor data. If you have a large amount of data which has low entropy, then compression can get you a lot. For example, common English text, has on average one bit of information per character. So you can compress an 80-column line of text (like this sentence) to 80 bits.

In this case, you take a data series, compress it, break the compressed data into packet-sized chunks, and send those chunks. The paper mostly explores the efficiency of different compression algorithms.

3 Assignment

The assignment is to implement these two approaches and compare how they improve energy efficiency. The basic measure is application-level throughput: how many bits per second of data can each approach deliver to another node?

When there are no bit errors, the CRC approach is overhead and less efficient than standard data communication. So you need to test these approaches in a network where there are bit errors. There are two approaches to doing this:

1. Have the radio transmit packets at a very low power with the command `CC2420Packet.setPower`. Move the nodes until they are far enough that they lose packets.
2. Make the radio operate on the same frequency as a WiFi (802.11b) card and start a lot of 802.11 traffic. Figure 1 shows the channel overlap between the two technologies and the effect this has on RF interference.

The first approach is a bit easier, but will make repeatable experiments very difficult. You’re essentially trying to put the nodes at the edge of RF sensitivity, and that means that if anything changes slightly they might move a bit to one side or another of the edge. The second approach requires that you have two computers with 802.11 and that you set them up appropriately, but will have more repeatable results.

The trick with creating 802.11 interference is that it has to be reasonably common. Downloading a file from the web won’t be sufficient, as you’ll be using something like 1% of the channel. You want to put both computers into ad-hoc mode on the right 802.11 channel, set up file sharing between them, and transfer a large file.

4 The Code

TEP 116 and Lesson 3 in the tutorials should get you started with packet communication. TEP 111 describes the TinyOS packet format. In addition to the basic, AM-level communication, you are going to need to do some radio chip specific things. For example, the AM layer does not provide functions to set the RF power, but the CC2420 radio does (the reason for this distinction is that setting the RF power is a very chip-specific thing to do correctly).

4.1 Hardware CRC

When implementing multiple-CRC packet protocol, you need to tell the radio to stop checking CRCs. Otherwise, it will discard packets that do not pass CRC, which defeats the purpose of your approach. To do this, you want to make a copy of the file `tos/chips/cc2420/CC2420ReceiveP.nc` in your application directory. **Be sure that you do not directly modify the file in chips/, and that you do not use your modified file when evaluating compression.** The line that you care about is this:

```
if ( ( buf[ length ] >> 7 ) && rx_buf ) {
```

This line tells the receive path to drop packets which do not pass a CRC check or if there is no receive buffer. The first predicate is examining the CRC bit in the packet. The radio automatically embeds a CRC in every packet on transmission, whether or not you have the receiver check CRCs. To have the receive path not check CRCs, all you need to do is ignore it:

```
if ( rx_buf ) {
```

4.2 Packet Length

Computing incremental CRCs is not very helpful when your packets are very short. You should increase the TinyOS packet payload size from 28 bytes to 100 bytes. You can do this by passing a parameter to `ncc` when it builds your application. In your Makefile, before the “includes” line, add this:

```
CFLAGS += -DTOSH_DATA_LENGTH=100
```

This defines the variable `TOSH_DATA_LENGTH` to be 100. The file `tos/types/message.h` looks like this:

```
#ifndef TOSH_DATA_LENGTH
#define TOSH_DATA_LENGTH 28
#endif

#ifndef TOS_BCAST_ADDR
#define TOS_BCAST_ADDR 0xFFFF
#endif

typedef nx_struct message_t {
    nx_uint8_t header[sizeof(message_header_t)];
    nx_uint8_t data[TOSH_DATA_LENGTH];
    nx_uint8_t footer[sizeof(message_footer_t)];
    nx_uint8_t metadata[sizeof(message_metadata_t)];
} message_t;
```

4.3 Data

This assignment require that you have data to compress and send. The choice of what data you use is an aspect of your experimental design. Sampling the light sensor, is reasonably, but will have relatively low entropy. Using the Random interface, in contrast, will have high entropy.

5 Writeup

The handin for this assignment is a 4-page paper, prepared in standard ACM conference style. The ACM has document templates you can use. The paper should have an abstract, introduction, related work, the whole deal.

Your paper should go something like this:

Introduction: Generally, an introduction serves three purposes. First, it tells the reader what problem you are investigating. Second, it articulates why this is important. Third, it summarizes why the paper makes a contribution and is therefore worthwhile to read.

Prior Work: This section usually comes after the introduction or at the end. Since this paper is comparing two proposed approaches, it is better to put it early. This section needs to summarize the two approaches.

The Middle: The center of the paper is your contribution. How you structure this is entirely to you. One way is to describe your experimental methodology, then present results. Or you can stitch them together. You choose which compression algorithms, packet sizes, and data sets to evaluate.

Conclusion: Here you should summarize your results and comment on their implications. I.e., “clearly you want to compress, unless you have random data.” There is of course more that you could do: state what you think would be the most useful areas of future work.

6 Handing In

The handin for this assignment is your four-page report. The report must follow ACM formatting guidelines and must be a PDF. In the email, please say which person in the group implemented and evaluated which approach.

7 Grading Criteria

Your handin will be graded on the following criteria:

- **Experimental design and explanation (60%):** How did you evaluate the tradeoffs of the two approaches? Does the resulting data provide meaningful evidence? For example, you should run multiple trials. Given the vagaries of wireless communication, running only a single trial makes it impossible to determine whether the resulting data is indicative of the common behavior of the approach or is an outlier. Similarly, *how* you measure things is important: turning on an LED when a transfer completes and using a stopwatch is much less precise than logging and timestamping with a laptop/PC. Finally, in order to give the reader confidence in your results, you need to clearly explain how you performed the experiments.
- **Observations (30%):** Based on your data, what conclusions can you draw? Given how much time you have spent on the problem, you probably have observations which might not be apparent to a reader. The more that these observations are substantiated by quantitative results, the better.
- **Prose (10%):** If a paper is badly or sloppily written, then it is hard to understand and suggests that the experiments were sloppy as well. First off, there should be no grammatical or spelling errors. You’re not going to be greatly penalized for minor typographical errors, but it should be clear that you’ve read the document a few times and have run it through a spell checker. Think of the criteria and expectations you have for a paper you read for class, and follow them.