

Assignment 4, CSci 330, Fall 2006

Due: Nov 3, 2:10pm. Value: 40 pts.

From the course Web page you can download a Java program designed for simulating network protocols. The program consists of the following classes.

Wire simulates a wire that transmits frames, with some random noise occurring during the transmission. Data sent down the wire is guaranteed to be received at the other end, with no bits dropped or inserted; the only effect of noise is to flip bits. The program creates two `Wire` instances — one for sending from a notional computer *A* to another hypothetical computer *B*, and another for sending back from *B* to *A*. This wire corresponds to the physical layer. (You will probably not need to examine this class.)

DataLinkLayer handles the protocol for transmitting packets over the wire. At present, this protocol simply transmits packets as unmolested frames — but of course this means there are errors in the transmission. The program creates two instances of this class, one for *A* and one for *B*.

TransferApplicationLayer attempts to send randomly generated text from one notional computer to another, using the data link layers at the respective computers. It displays errors when the received text does not match the original text. The communication is entirely one-way at this layer. (You will probably not need to examine this class.)

TextGen is used by `TransferApplicationLayer` to generate text with a character distribution similar to that used by the human language that many people call English. (You will definitely not need to examine this class.)

Main contains the main method, which sets up the various layers, kicks off the transmission, and prints statistics after the transmission completes.

BitStream is a utility class for your convenience. It is useful for manipulating an array of bytes as a sequence of bits.

Your job is to modify `DataLinkLayer` so that it transmits data flawlessly (or as near flawlessly as possible). You should use CRC (Cyclic Redundancy Check) for your error detection. Recall from class that our CRC algorithm is the following:¹

```
poly = a pre-selected bit sequence
mask = bit mask with one 1 bit wherever poly's most-significant 1 bit is
rem = 0
for each bit b of frame:
    if rem & mask == mask:
        rem = ((rem ^ poly) << 1) | (1 - b)
    else:
        rem = (rem << 1) | b
return rem
```

As a choice for the polynomial `poly`, I recommend using `0xBAAD`, which will result in a 16-bit remainder.²

¹You'll find alternative formulations in other sources. Use mine.

²Koopman and Chakravarty introduced this bit sequence in their 2004 article, "Cyclic redundancy code (CRC) polynomial selection for embedded networks" (*International Conference on Dependable Systems and Networks*, 2004). For long frames, it appears to be the best order-16 polynomial — and it's easy to remember, too!

An earlier article by Koopman investigates 32-bit polynomials and suggests `0xBA0DC66B` as an excellent choice (Koopman, "32-bit cyclic redundancy codes for Internet applications," *International Conference on Dependable Systems and Networks*, 2002). The standard 32-bit sequence used in IEEE 802 protocols is `0x82608EDB`.

Also, after doing the above, you should experiment to determine roughly the optimum frame size if the noise occurs with independent probability of 10^{-6} per bit (a relatively clean stream), and also if the noise probability is 0.1% per bit (a very noisy stream). Doing this is simply a matter of running your program with various configurations of the constants appearing at the top of `Main`.³ (Actually, the constant in `Main` refers to the packet size rather than the frame size; but since your data link layer won't break packets into frames, this will be fine.) Write a brief paper (at most one page) discussing the relationship between frame size and transmission overhead. Your paper should provide your numerical results and explain why both tiny frames and large frames are poor choices.

For this assignment, you do not need particularly to worry about writing a particularly efficient protocol for communication between the two ends.

To submit your solution, you should e-mail me your Java source for `DataLinkLayer`, and you should submit your typed paper to me physically.

³Please don't waste your time trying to narrow in onto the "optimum" frame size. Trying 10 different values for frame size should be good enough.