**RDT: the Python version**

This code was contributed by Eric Eide, who was teaching the introductory computer science course at the University of Utah in the Spring of 2022. Thank you so much Eric for contributing this. You rock! Here are his notes:

In zip file, rdtsim\_python\_lab\_only.zip please find:

 rdtsim.py --- the simulator with stubs

 solution-ab.py --- a solution for alternating-bit[[1]](#footnote-1)

 solution-gbn.py --- a solution for go-back-N

This is Python 3 code. It needs Python 3.6+ because I used "f-strings." Python 3.6 was released in Dec 2016. Some things/changes you'll notice[[2]](#footnote-2):

* The simulation parameters are specified via command-line options. The architecture of the code is such that an autograder script can specify the parameters programmatically.
* There is a `seqnum\_limit` parameter (-z) that sets the allowable range of sequence numbers. E.g., "-z 4" means that one can use 0-3. One can use this to enforce AB sequence numbers (-z 2), but more generally, I implemented this just so that students would have to deal with it . If an instructor doesn't want students to deal with this, they could set the limit very high.
* There is a random seed argument (-s) so that students can get repeatable results (and the autograder, too). I removed your check-the-PRNG code because Python provides a good and portable PRNG.
* The simulator automatically prints the configuration at the start and the result stats at the end. I did this because (1) it seemed helpful to students and tedious for them to do for themselves, but mostly because (2) I needed a stats API for my autograder anyway, so... There is an API for an autograder to get stats.
* One can specify callbacks for `to\_layer5` events. These are unused in the simulator and in the solutions. My autograder uses these hooks to check that the message delivered at the receiver are correct.
* The corruption algorithm is different, but this doesn't much matter in practice.

Let me mention that the solutions are "existence proofs" and are not well-commented. They are not hard to understand, I think, but if I were going to give these solutions to students (which I am not), I would add more comments, and I would probably redo the AB solution altogether (because I did it in a "weird way"). The timeouts in the solutions are "OK" but I didn't spend much time tuning them.

1. Note that solutions are available only to instructors. Instructors can contact kurose@cs.umass.edu for solutions. [↑](#footnote-ref-1)
2. Eric is referring to changes from the original C version. [↑](#footnote-ref-2)