### Switching to TCP

In Project 1 you wrote a UDP server allowing a file to be transferred to clients in 32 KiB sections. While files should be transferred successfully when running on localhost or across a local area network, packets may be lost when the network is heavily loaded or the connection requires several hops. To address the possibility of packet loss, we will switch to the TCP protocol.

### Server

SectionServer.py implements a TCP variant of the application protocol defined in Project 1, with the following additions and clarifications:

1. The TCP connection will be closed after responding to a LIST request.
2. The TCP connection will remain open after responding to SECTION requests in order to allow additional sections to be requested.
3. Erroneous requests will result in a response beginning with the word ERROR: followed by a human-readable string.

### Client

Implement a TCP version of SectionClient.py and verify that it can download a file successfully from the server.

#### Reliability

Unfortunately, while the transport protocol may be reliable, the hosts claiming to implement the application protocol may not be. While responses to LIST requests can be trusted, requests to download sections may not always succeed. Your client should recover from these failures.

### Test Environment

You may use any platform for development, but note that per the Syllabus the test environment for projects in this course is a Tuffix VM with Python 3.6.x.

### Tips

* Read the Socket Programming [HOWTO](https://docs.python.org/3/howto/sockets.html#using-a-socket).
* When you’re ready to start testing your code, consider setting UNRELIABLE to False.
* When you’re sure you can download a file successfully, don’t forget to change it back to True.
* SectionServer.py can run on more than one port. Consider what that might mean for Project 3.