**Project 3**: Reliable File Transfer Over UDP  
  
*Type of Project*: Individual or Pair  
*Deadline*: [deadline]  
*Points*: 25 points  
  
***Submission Guidelines***: Submit through nike.cs.uga.edu, as usual (course dir cs4760). Name the directory project as "**LastName\_FirstName**-udp\_reliable\_transfer" (e.g., "Perdisci\_Roberto-udp\_reliable\_transfer"). Submit the source code, README, and a document that describes **your client-server communication protocol** (e.g., format of the packet header, in what cases you send ACKs or NAKs, what is the length of time-out, etc.). You may create as many classes as needed. However, your client class must be called **UrftClient** and the server must be called **UrftServer**.  
  
Example structure:

Perdisci\_Roberto-udp\_reliable\_transfer | +--- Protocol.txt | +--- README | +--- UrftClient.java | +--- UrftProtocol.java (class for common code) | +--- UrftServer.java | +--- ... (other .java files)

NOTE: project submissions that do not follow the guidelines risk to be discarded without consideration (i.e., 0 points).  
***Project Description***: In this project, you are required to **design a reliable file transfer protocol** and **write a client application and a server application** that communicate with each other using UDP use a **stop-and-wait approach**. In particular, the client needs to contact the server, pass the file name to be uploaded to the server, and send the file to the server. The client-server communication **must withstand packet loss**. When stated, UrftServer will wait for a client to connect. **Once a client has successfully transfered a file, UrftServer will exit**.  
  
NOTE: 

* Max Datagram Size = 512 bytes. This is the maximum length of the UDP payload that you should use.
* Max Time-out = 1sec.

***Testing Your Code***  
To test your code we will create two directories: "client\_dir" and "server\_dir". These directories will be the source and destination directories respectively.   
  
To simulate packet loss, we will test your software using a "transparent proxy" [[amd64](http://www.cs.uga.edu/~perdisci/CSCI4760-S12/Assignments/Project3_files/urft-proxy_amd64), [i386](http://www.cs.uga.edu/~perdisci/CSCI4760-S12/Assignments/Project3_files/urft-proxy_i386)]. The transparent proxy will probabilistically drop a packet.   
An example of how we will run your code is reported below:

### This assumes that you have compiled your submission directory with ### $ javac \* ### and that the client\_dir and server\_dir directories exist. ### [ Terminal 1 ] ### Destination directories $ cd server\_dir ### java -cp .. UrftServer <server port> $ java -cp .. UrftServer 20000 ### [ Terminal 2 ] ### urft-proxy <loss probability> <proxy port> <server ip> <server port> $ ./urft-proxy 0.10 10000 127.0.0.1 20000 ### [ Terminal 3 ] ### Destination directories $ cd server\_dir ### urft-client <proxy ip> <proxy port> <file name> $ java -cp .. UrftClient 127.0.0.1 10000 test\_file1

In this example, the file "test\_file1" will be split into datagrams and sent to the server, which will store it in the "server\_dir" directory. You can then compare the md5sum of "client\_dir/test\_file1" and "server\_dir/test\_file1" to make sure they match!  
  
***Mailing List***: If you have questions about the projects, the best place to ask is the course mailing list.  
  
***Project Evaluation***  
We will run your program to transfer 5 different files, configuring the server to run with different loss probabilities. For each of the 5 runs, you will be assigned 5 points if the md5sum of the final uploaded file matches the md5sum of the original file on the client. Therefore, you will get max points if all the md5s match correctly.  
  
The protocol specification document submitted along the code will be taken into account. Unsatisfactory protocol specifications will negatively impact the your project grade up to 5 points. For example, if you do not submit the specifications, or if they are not clearly stated, you may receive as low as 20 points even if the code works.