# Lab 3 – TCP/UDP File Transfer utility

## Overview

We have decided to use Python for the implementation of socket transfer. Our server sends and receive data using both TCP and UDP protocols. TCP is primarily used for transferring of metadata and UDP is primarily used for transferring of data chunks (pieces of files).

## Approach

We have decided to transfer the files using two separate streams, transferring the metadata using TCP and transferring the actual file pieces using UDP. Since metadata about the file, such as file size and checksum, need to be preserved, we’ve decided to send that through the TCP protocol. We decided to use Python’s file read function to read a certain blocksize and transfer the amount that’s read.

A summary of the metadata information is shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| TCP - metadata | | UDP - data | |
| Client | **Server** | **Client** | **Server** |
| File name | Resend data information | Send data chunks | Receive data chunks |
| Total # of blocks |  | Block Index (marks which block # is being transferred) |  |
| Checksum of file |  |  |
| Checksum of each block |  |  |
| Total Size of file |  |  |
| Size of each block |  |  |

## Functions

generateMetaData – This function will take in a file and generate the following metadata information:

* Size

## Modules/Libraries

In our project, we made use of the following public python libraries

* socket – this is the low-level networking interface for socket servers and clients
* os – contains portable tools to help the user use operating system dependent functionality, such as getting the size of the file
* hashlib – we used md5 checksum function in this security interface to ensure that files sent and received matched the same hashed md5 checksum