File Copy Protocol Design Specification
By Dylan Hoffmann & Lucas Campbell

- Network Protocol:
    - The network is composed of 3 types of packets:
        - Directory Pilot Packets (Type D)
        - File Pilot Packets (Type P)
        - File Data Packets (Type F)
    - Packets are generally sent in the above order.
    - Pilot packets communicate metadata from the client to the server
    - Data packets transfer file data from the client to the server with minimal basic identifying information
- Packet Specifications:
    - **Directory Pilot Packet (Type D)**
        - In the form: "D ####### HHHHHHHHHHHHHHHHHHH"
        - Where:
            - D is the packet type indicator for a Directory pilot packet (Always 1 byte)
            - # is the number of files in the directory (Up to 7 bytes = 9,999,999 files)
                - Always padded to 7 bytes
            - H is the SHA1 checksum of the directory (Always 20 bytes)
    - **File Pilot Packet (Type P)**
        - In the form : "P ####### IIIIIII HHHHHHHHHHHHHHHHHHHH F....."
        - Where:
            - F is the packet type indicator for a file Pilot packet (Always 1 byte)
            - # is the number of packets for the file == (file-size // 420) +1 (up to 7 bytes = 9,999,999 packets)
                - Always padded to 7 characters
            - I is the file ID (up to 7 bytes = 9,999,999 files)
                - Always padded to 7 bytes
            - H is the SHA1 hash of the file (Always 20 bytes)
            - F... is a variable length field for the file name (up to 470 bytes long)
    - **File Data Packet (Type F)**
        - In the form: "F ####### PPPPPPP D....."
        - Where:
            - F is the packet type indicator for a File data packet (Always 1 byte)
            - # is the packets number of this packet (Up to 7 bytes = 9,999,999 packets)
                - Always padded to 7 bytes
            - P is the file ID (Up to 7 bytes = 9,999,999 files)
            - D... is a variable length field for the file data (up to 480 bytes long)
- Given the above specifications the constraints of this system become apparent:
    - Overflow can occur in directories containing more than 9,999,999 files, or files larger than 4.8 Gb.
        - However, both of those overflows seem unlikely (though certainly not impossible) and this system can currently handle, assuming roughly uniform file sizes, almost

48 Petabytes of data within the existing confines of the protocol. (9,999,999 files x 9,999,999 packets x 480b)

  - Each File copy attempt requires at least 2 + ((file_size // 480) + 1) packets which, while not egregious, under sufficiently terrible network conditions the requirement for pilot packets may become an apparent drawback.
- The benefits of this system come from the same feature which contributes to one of the drawbacks, because metadata is sent separately there is more space in the data packets for transmitting file data, which we suspect will allow us to need fewer packets, and allow our protocol to be faster than the alternative methods we considered.
- End-to-End check
  - Our implementation involves end-to-end checksum comparison checks on each file, as is suggested in the Assignment specification, however, we felt that that approach did not satisfy the intent of the end-to-end principle. As such, our program implements an end-to-end checksum comparison after all files have been “transferred” of the SHA1 checksum of the Target directory to the checksum of the source directory.
    - Additionally, we perform the asked-of end-to-end checksum on individual files.