# By


- **Abhishek Bapna**
  2018A7PS0184H
- **Ashna Swaika**
  2018A7PS0027H
- **Siddhant Kulkarni**
  2018A7PS0185H
- **Sravani Garapati**
  2018A7PS0097H
- **Vikram S Haritas**
  2018A7PS0302H


# Introduction

This is a much better implementation of the Selective Repeat algorithm which is highly multithreaded and removes the bottlenecks helping us reach best case transfer rates of over 424Mbps (tested on a file of size 76MB). Our implementation spawns 100 (Changeable) threads and is compatible with pure Selective Repeat Paradigm.
There are 2 requirements for our program
Multitimer : For multithreading pypi
TQDM: For rendering progress bars pypi
The biggest file sent had this statistic.

```
Time Elapsed 74.91866517066956   File Size: 1.18 GiB
Avg IN Speed 16.10 MiB/s
```

# Install Requirements

pip install -r requirements.txt
Execute the above command in the terminal where the current directory has requirements.txt file. Installing the above requirements.
The requirements are
tqdm==4.59.0 multitimer==0.3

## RUNNING SCRIPT

### Receiver Script

python3 sender.py ServerIP ServerPort PacketSize

ex
python3 sender.py 127.0.0.1 6060 1024
The above command Starts a socket bind to ip 127.0.0.1 and port 6060 it listens for clients and accepts file packets from them. Initially with a packet size of 1024 but later follows the packet size used by sender.py.

### Sender Script

python3 sender.py filename ServerIP ServerPort PacketSize

ex
python3 sender.py vid.m4v 127.0.0.1 6060 1024
The above command gets the file ‘vid.m4v’. And sends to a receiver on port 127.0.0.1 on port 6060 with a packet size of 1024B.

## IMPORT LIBRARY

### Receive file

from Protocol.protocol import receive_file
receive_file(SERVER, PORT, size)

ex
receive_file(‘127.0.0.1’, 6060, 1024)
a Receiver setup at ‘127.0.0.1’ and port 6060 packet size 1024

### Send File

from Protocol.protocol import send_file
send_file(filename, SERVER, PORT, size)

ex
send_file(‘vid.mp4’, ‘127.0.0.1’,6060, 1024)
a sender setup at ‘127.0.0.1’ and port 6060 packet size 1024

# Changes in Design Doc

Packet Structure
A packet has structure based on its implementation.
The different information is delimited by ‘;’ by default the packet has an integer embedded in the beginning and closed by a ‘;’.

1. Regular Packets have an index at the beginning followed by a ‘;’ followed by the data that is being sent (in bytes encoded by UTF-8 format) which is then once again followed by a ‘;’ to close the packet.
2. Special Packets
    a. These packets have the index as -1, they serve as special purpose packets, there are 4 different types of special packets:
        i. Hello: This packet is used to initialize the connection which is sent by the one who wants to send a packet to the server, it has multiple fields, the first being index -1 and the rest are:
            1. String “HELLO”: Indicates the HELLO message.
            2. Filename: This is what is used as the filename for the data received.
            3. Count: The number of Packets to be transfered.
            4. Window Size: Used for transfer of packets.
        ii. HelloBack: This packet is sent by the receiver to acknowledge the connection request sent by the sender. This packet contains the string “HELLOBACK”.
        iii. Close: This is sent by the sender when all packets are Acknowledged. This packet is sent by the sender to close the connection. This packet contains the string “CLOSE”.
        iv. CloseBack: This packet is sent by the receiver to acknowledge the close connection request sent by the sender. This packet contains the string “CLOSEBACK”.