Names:

Sajid Hussein	921481041
Behdad Feilizadeh	921484394
Youssef Qteishat	921490016

Python files:

sender_stop_and_wait.py	sender_tahoe.py	sender_custom.py
sender_fixed_sliding_window.py	sender_reno.py	…

Custom Algorithm (TCP Sharm El-Sheikh):

For our custom Algorithm, we created an iteration of TCP Vegas. The motivation behind the algorithm is to optimize throughput and run time by avoiding duplicates and time-outs, which differs from TCP Tahoe and TCP Reno.

Similar to TCP Tahoe and Reno, the initial ***cwnd*** (control window) is user-specified. However, instead of waiting for congestion to occur, the algorithm predicts it before it happens by comparing the expected and actual throughput each time a packet is sent (see equations below). We then take the difference between them and adjust the control window size accordingly. We do this by comparing the difference to the user-specified ***alpha*** and ***beta*** threshold values. In our case, we found that the following values produces the best performance metrics compared to TCP Tahoe and TCP Reno:

$cwnd = 20, \alpha = 2, \beta = 4$

The main difference between TCP Sharm El-Sheikh and TCP Vegas is that instead of working with a continuous stream of packets directly from the MP3 file when adjusting the control window, we are storing the file's packets into a ***messages*** list and reading the packets into a ***window*** list with the same size as our *cwnd* value. We then iterate over the *window* list to perform our custom procedure on each packet. After which we set ***window*** to the next set of packets from ***messages*** depending on the new ***cwnd*** value and perform the procedure again. After all packets have been read, we exit the file. In the event of a duplicate ACK, we perform a fast retransmit of the packet, however it's unlikely to occur.

Metrics:

Algorithm	Throughput	SD-Throughput	Per-packet delay	SD-ppd	Performance metric	SD-performance
Stop & Wait	18564.35	77.55	0.11	1.39	172783.23	1503.71
Fixed Window	53782.97	5520.37	1.92	0.24	28743.94	5270.12
TCP Tahoe	83218.74	5444.17	0.54	0.05	155285.33	13010.14
TCP Reno	82805.33	4577.31	0.54	0.06	155738.20	17846.39
TCP Sharm El-Sheikh	78603.30	2680.20	0.15	0.003	530142.78	21013.06

Equations

$$Expected = \frac{cwnd}{Base\ RTT}$$

$$Actual = \frac{cwnd}{Current\ RTT}$$

$$Base\ RTT = RTT\ of\ first\ packet\ of\ the\ file,\ otherwise\ Min(Base\ RTT,\ Current\ RTT)$$

$$diff = (Expected - Actual) * Base\ RTT$$

$$\alpha = 2,\ \beta = 4$$

$$cwnd_{initial} = 20,\ cwnd_{new} = \{cwnd\ +=\ 1\ when\ diff < \alpha;\ cwnd\ -=\ 1\ when\ diff > \beta;\ otherwise\ cwnd = cwnd\}$$