Report

In sender.py:

Data structure design:

Create sender socket -> two-way handshake -> separate files according to mss -> Sender Thread and Receiver Thread -> one-directional connection termination.
Sender Thread:
Only if difference between last sequence number sent and last sequence number received is less than max_window
-> transmit segments and start a timer to determine whether a segment is timeout
-> check whether have get a receiving sequence number from receiver.
(-> if timeout, retransmit the segment and restart a timer)
-> if at the end, break loop and exit the thread
Receiver Thread:
Only if the timer is started.
-> Check whether receive the expected sequence number from receiver
(-> if get expected number, then change expected number)
(-> if get the same one for three times, then fast retransmit the oldest segment)
-> if at the end, break loop and exit the thread

In Receiver.py:

Data structure design:

Create receiver socket -> Loop to receive message from sender: define a function called segment loss to check whether message should be dropped -> one-directional connection termination.
Loop design: (sequence of determining message below)
1. if the type is reset, close socket and exit system. Otherwise, go on.
2. determine whether the segment is dropped in the way of forward by the function, segment loss, or not. If dropped: only record the time of dropping. Otherwise, go on.
3. determine the type of segment received: if the type is SYN, use the function to judge if the SYNACK will be dropped in the reverse way.
If the type is data, check whether it is the expected data -> if the data is expected, change the expected data and check buffer whether there are segments in the out of order to write file and change the expected data, then send back sequence number expected in the sender-> if the data is not expected, record it and put it in the buffer, then send back the original expected sequence number and store it in the buffer.
If the type is FIN, create another thread to implement a timer to close socket and system by a function called threading.timer and then if FINACK is sent back, cancel the timer and break the loop.