**Problem:**

Develop a UDP-based client-server socket program for transferring a large message. Here, the message transmitted from the client to server is read from a large file. The message is split into short data-units (DUs) which are sent and acknowledged in batches of size 1, 2, and 3 DUs. The sender sends one DU, waits for an acknowledgment (ACK); sends two DUs, waits for an ACK; sends three DUs, waits for an ACK; and repeats the above procedure until the entire file is sent. The receiver sends an ACK after receiving a DU; sends next ACK after receiving two DUs; sends next ACK after receiving three DUs; and repeats the above procedure. Verify if the file has been sent completely and correctly by comparing the received file with the original file. Measure the message transfer time and throughput for various sizes of data-units and compare it with the stop-and-wait protocol where the batch size is always fixed to be 1. Choose appropriate values for “data unit size” and measure the performance. Repeat the experiment several times and plot the average values in a report with a brief description of results, assumptions made, etc.

Include the following performance figures in your report: 1) Transfer time vs data unit size ( one curve for this problem and one curve for stop-and-wait ) 2) Throughput vs data unit size ( one curve for this problem and one curve for stop-and-wait )

**Assumptions:**

1. Results may vary due to the performance of the virtual machine. Under different environments (different virtual machines), results might differ.
2. No other external programs are using the same port number
3. No timeout
4. All the packets sent have no issues. There are no lost packets/acks, ARQ error and retransmissions.
5. No multiple clients. Only 1 client is transmitting till the entire file has been transmitted and received.
6. Maximum file size sent is 60000 bytes.
7. Network connection is consistent.
8. Same conditions was mimicked for results (e.g: same file of 59792 bytes used (myFile.txt text x 2)

**Performance (Transfer time vs data unit size)**

1. The transfer time for stop and wait is larger as compared to variable DU approach probably because of increased acknowledgements for stop and wait.
2. As the data unit size increases, the transfer time decreases at a decreasing unit.
3. Total transfer time is inversely proportional to data unit size
4. Transfer time is computed as transmission time + 2 x propagation time + queueing delay. Queuing delay can be ignored and ack transmission time can be ignored.
5. For stop and wait, the number of acknowledgements is file size over data unit size.
6. Theoretically, if data unit size = 0, transfer time is infinite.
7. So, as data unit size increases, number of times to send and receive packets decreases proportionally.
8. Assuming transmission time is same (since same file is used -> same file size) and given the above, total transfer time is inversely proportional to data unit size

**Performance (Throughput vs data unit size)**

1. As the data unit size increases, the throughput increases at a decreasing rate. This is probably because throughput is measured by taking the file size over the total transfer time. For a given similar file size (59792 bytes in this case), since we saw from above that the total transfer time decreases at a decreasing rate, the throughput is seen increasing at a decreasing rate.
2. For variable DU window size , as the data unit size increases, more bytes are sent for a unit of time.
3. For variable DU window size, since lesser ack packets are sent, the link utilization is higher. Throughput is also higher as compared to stop and wait approach.