Anton Krotenok (NETID: ak1847)

Kevin George (NETID: kmg328)

**\*PLEASE READ\***

From line 280 to 296 (commented as “START OF USERNAME” and “END OF USERNAME”) in the receiver.py file, to test step 5 these lines need to be commented in. Testing step 5 had some problems and it doesn’t work fully. So, in order to not cause any problems with the testing of steps 3 and 4, we’ve commented out step 5 (to be commented back in for testing it).

1. Team details: Clearly state the names and netids of your team members (there are 2 of you).

**Anton Krotenok (NETID: ak1847)**

**Kevin George (NETID: kmg328)**

2. Collaboration: Who did you collaborate with on this project? What resources and references did you consult? Please also specify on what aspect of the project you collaborated or consulted.

**Both partners utilized past knowledge from classes including Computer Architecture, Software Methodology, and Principles of Information and Data Management.**

3. Is there any portion of your code that does not work as required in the description above?

Please explain.

**There is no portion of our code that does not work as required in the description above.**

4. Did you encounter any difficulties? If so, explain.

**Please read the “\*PLEASE READ\*” on the top of the page for further details.**

5. Describe two technical observations or facts you learned while working on this project. Please answer in specific and precise terms. Your observations could relate to topics including reliable data delivery in general, your implementation of it in this project, reliable delivery in TCP, using UDP sockets, or other topics that are relevant to your software and implementation in this project. Please ensure your responses are clear, specific, and technical.

**1. UDP sockets don’t always provide a reliable interface over lossy networks. Most common in media-heavy entertainment (eg. watching videos, gaming), users who experience over 2% packet loss are expected to find noticeable problems. Packets that are resent to a client cause “lag”, a delay between the action of the user and the reaction of the server supporting the task. The “timeout” value of the protocol dictates the extent to which data is retransmitted. It is typically longer than the RTT value for the purpose of having greater control over the infrastructure. Too short of a timeout value cause unnecessary retransmission whole too long of a timeout value causes a slow reaction to the segment loss.**

**2. transmit\_one() is called once after a timeout. Since we are using selective repeat, this is used to ensure that the receiver program is buffering out-of-order data to reduce unnecessary transmissions. A pipelined sner has a distinct timeout for each sequence number. Noticeably, the select() function implements a timeout on the entire socket. We see that the difference between timeout per socket and timeout per sequence numbers did not matter in “stopandwait” due to the fact that sequence numbers in flight will either be ACKed or not. A pipelined program is special because the select() call prevents a single lost packet from causing a timeout.**