Distributed Systems Exercise 1
Matt Richard, Ron Boger

Our distributed system used the **selective repeat** protocol as presented in class. At a very high level, the selective repeat protocol uses a **sliding window** to send multiple packets at a time. The receiver then acknowledges these packets using a **cumulative acknowledgment** and a **negative acknowledgment**. The cumulative acknowledgement indicates that all packets up to and including its value have been acknowledged by the receive, whereas the negative acknowledgment indicates that a packet at a given index has not been received. We will transfer files over both the UDP and TCP internet protocols, and aimed to make our approach as generalizable as possible.

In an include file, both the sender and receiver have access to:

- WINDOW_SIZE for the selective repeat protocol
- SEQ_SIZE corresponding to the number of bytes of the file sent per packet
- PORT of 10170 (ron's assigned port on the ugrad machine)
- The packet the sender and receiver both send

At a lower level, the sender contains/sends the following:

- The file itself
- The host name/IP of the receiver it wants to send to
- A packet struct, which contains an integer sequence number (indicating where in the file we are), the number of bytes to read, and an array of bytes (the actual data we want to transfer over the network). **For future reference, we call this struct "*dataMessage*"**

And the receiver contains/sends the following:

- A queue of identifiers of the senders, which contains a sender address and destination file name
- An acknowledgement struct which contains an integer cumulative acknowledgment and an array of indices of the negative acknowledgements, both of which we defined at the top of this design file. **For future reference, we call this struct "*AckPacket*"**
- A byte window buffer to receive the file, which gets dynamically cleared

Sending files is handled in the following manner:

- Initially, the sender will send one packet to the receiver having sequence number -1 and the filename we seek to name it on the other machine. This leads into 2 cases:
    - The receiver realizes it's not busy and starts accepting packets from the sender
    - The receiver adds the sender to a queue of senders (assuming the sender is not already in the queue) and alerts the sender that it is busy by sending a packet with cAck of -2. If the sender doesn't get receiver busy message, it will keep sending until the receiver sends confirmation. Once the receiver is ready, it will send a packet to the sender with cAck of -1 until the sender sends the first message.
- Once the receiver is ready, the sender begins sending the actual data

    - This follows the typical selective repeat protocol - we send all packets within the window and receive an AckPacket from the receiver. With the information from the AckPacket, we choose to shift the window within the file sequence based on the cAck received and resend nAcked Packets.
    - A combined nAck and cAck message from the receiver is sent either when it receives the a full window from the sender or when it times out waiting for the receiver to send the next message.
- To disconnect, we do the following:
    - The sender sends a message with sequence number -2 to identify itself. The receiver will still respond to disconnect messages from the sender even if it is not serving the sender, so we continually send a disconnect request from the sender until we get confirmation from the receiver.
    - Close the file reader for reading in bytes from the source file

To be concise, assume operations for the receiver already mentioned in the function of the sender, as their functions are intimately tied. The receiver at a lower level functions as:.

- When the receiver processes a sender:
    - It receives a dataMessage and writes to the corresponding window slot in the destination file
    - Slides the window as far as necessary, writing to disk any packets that we slide past
    - Sends an AckPacket with cAcks and nAcks based on the state of the packets received within each individual window
    - On timeout, we resend the last AckPacket we sent to the receiver
- When the receiver gets a connection request but is busy, it queues the sender and sends it a “busy message”, corresponding the a AckPacket with cAck of -2
- To disconnect a sender:
    - We receive the disconnect request from the sender
    - The state of the window to hold and write data is reset as the file is already written
    - Send a response to the sender attempting to disconnect, regardless if it is the sender we are actually serving