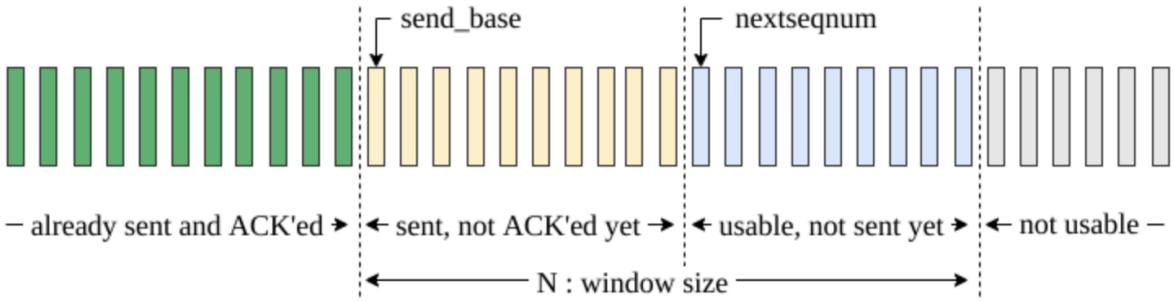
So, actually I just implement the GBN for assignment 2

So, for sender part:



First of all, the sender starts by sending the first frame. Initially, and . While there are more packets to send and the is smaller than the ; the sender sends the packet pointed by the pointer and then increments the .
Meanwhile, the is incremented after receiving acknowledgment packets from the receiver. The reception of duplicate ACK messages does not trigger any mechanism.
There is a single timer for the whole sending window, which measures the timeout for the packet at the . Therefore, if a timeout occurs, the sender restarts the timer and re-transmits all the packets in the sending window starting from .
To summarize, we can represent the sender's algorithm with the following pseudocode:

Algorithm 1: Go-Back-N Sender

```
Function Sender is
    send_base ← 0;
    nextseqnum ← 0;
    while True do
        if nextseqnum < send_base + N then
            send packet nextseqnum;
            nextseqnum ← nextseqnum + 1;
        end
        if receive ACK n then
            send_base ← n + 1;
            if send_base == nextseqnum then
                stop timer;
            else
                start timer;
            end
        end
        if timeout then
            start timer;
            send packet send_base;
            send packet send_base + 1;
            ...
            send packet nextseqnum − 1;
        end
    end
end
```

For receiver:
The receiver only keeps track of the expected sequence number to receive next: .
There is no receiver buffer; out of order packets are simply discarded. Similarly, corrupted packets are also silently discarded.
It always sends the acknowledgment for the last in-order packet received upon reception of a new packet (successfully or unsuccessfully). As a result, it will generate duplicate acknowledgment messages if something goes wrong.
As a summary, the pseudocode for the receiver's algorithm is:

Algorithm 2: Go-Back-N Receiver

```
function Receiver is
    nextseqnum ← 0;
    while True do
        if A packet is received then
            if The received packet is not corrupted and
             sequence_number == nextseqnum then
                deliver the data to the upper layer;
                send ACK nextseqnum;
                nextseqnum ← nextseqnum + 1;
            else
                /* If the packet is corrupted or out of
                   order, simply drop it                  */
                send ACK nextseqnum − 1;
            end
        else
    end
end
```