CN homework#3 report

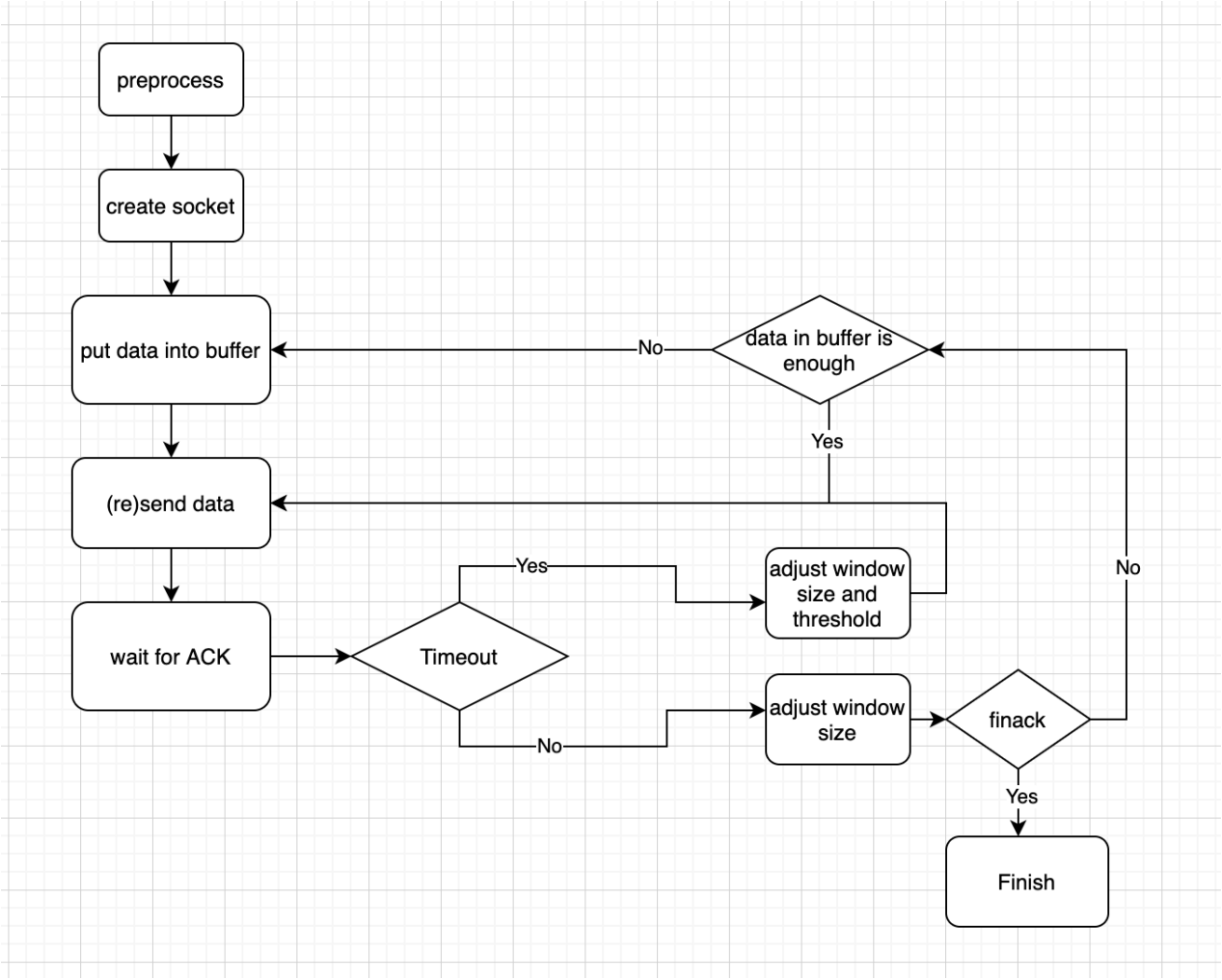
b07902076
資工三 許世儒

How to execute programs

```
# compile
make

# execute server
./server <svr port> <agent IP> <agent port> <file path>

# execute agent
./agent <svr IP> <recv IP> <srv port> <agent port> \
 <recv port> <loss rate>

 # execute receiver
./receiver <recv port> <agent IP> <agent port>
```

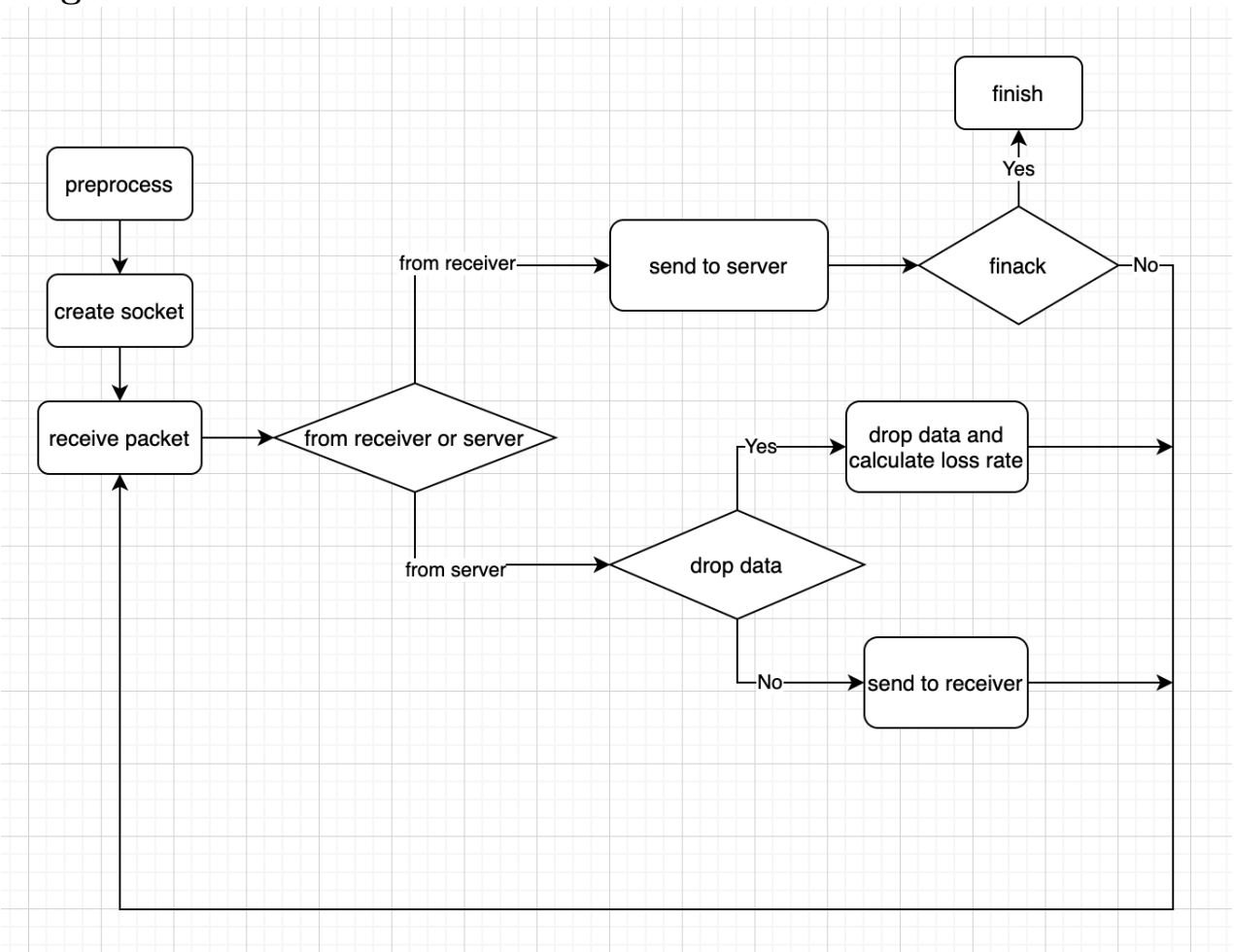
Program structure

server:



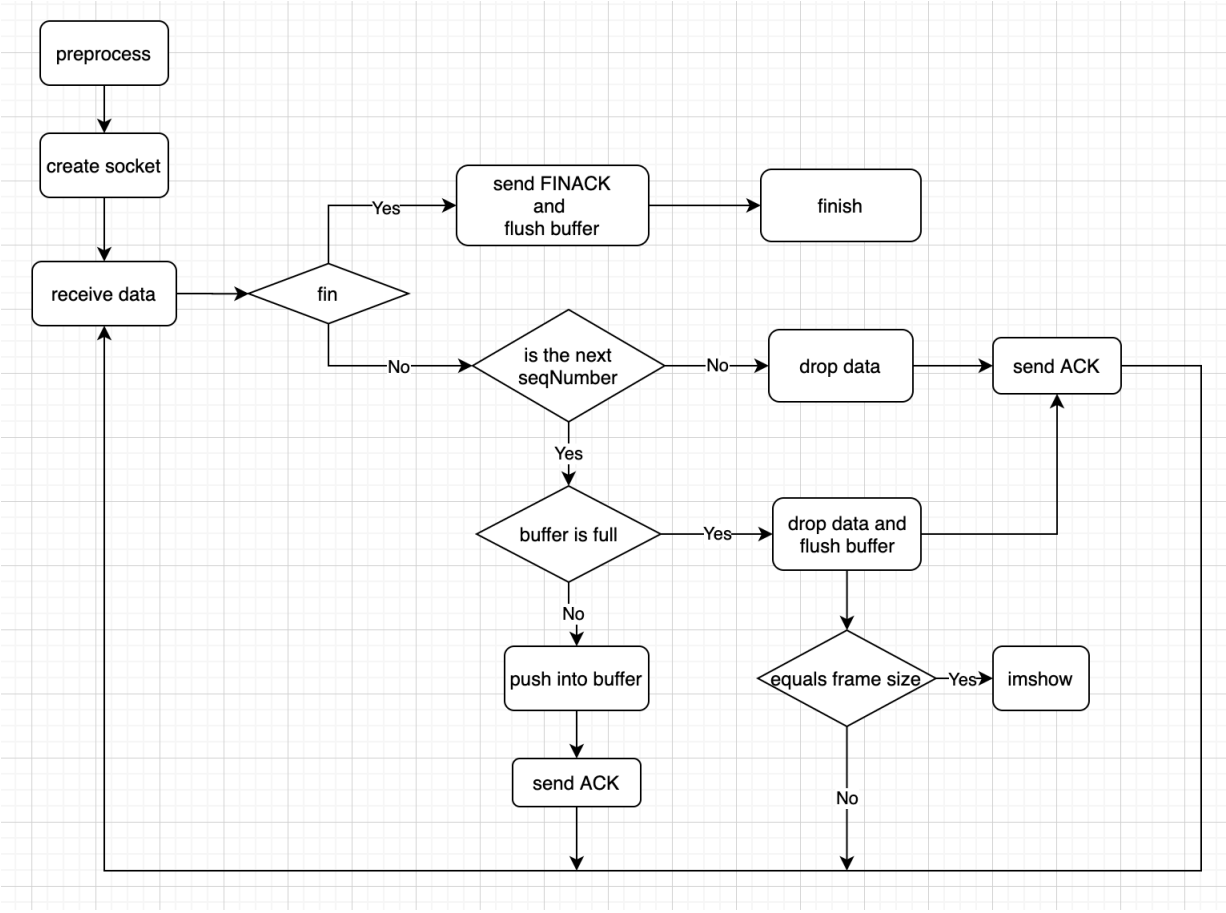
After creating socket, I initialized the image server and get the information of the video. I used a data structure "deque" as the buffer for saving packets which were ready to send. Then, the first packet, I sent the height, width and eleSize to the receiver. After that, there is a long while loop for critical jobs. First, I checked whether the buffer size is smaller than window size. If it did, then I got data from imgServer and put them into buffer. Otherwise, I sent data from the buffer to the agent and then receive ack from agent with setting timeout to be 50000(μs). If timeout happened, then I resized the window size and threshold and resend data. If it didn't, then I popped data from the buffer if the ackNumber equaled to the seqNumber. This loop would be stopped when the video finished, sent fin to the agent and received finack from the agent.

agent:



For agent, I used the code that TA provided. After creating socket, I received packets and determined who had sent the packet. If the packet was from receiver, then I checked whether the ack in the header of the packet was set. Then, it would send the packet to the server and if the packet was FINACK then the program would be terminated. On the other hand, if the packet was from the server, then I checked whether the ack in the header of the packet was not set. Then, it would randomly drop data based on the random number. Finally, if the data was not dropped then it would be sent to receiver, otherwise, the loss rate would be calculated.

receiver:



After creating socket, it kept receiving data until the fin was received and sent FINACK back to agent and flush buffer. Otherwise, it would check whether the sequence number of the received packet equals the next sequence number it required. If not, it would drop the data and send ACK. If it did, then it would check whether the buffer was full. If the buffer was full, then it would drop all the other packets in that round and sent ACKs then it would flush the buffer and check whether the frame of video could be played. When the buffer was not full, then it would push the data into the buffer and sent ACK back to the agent.