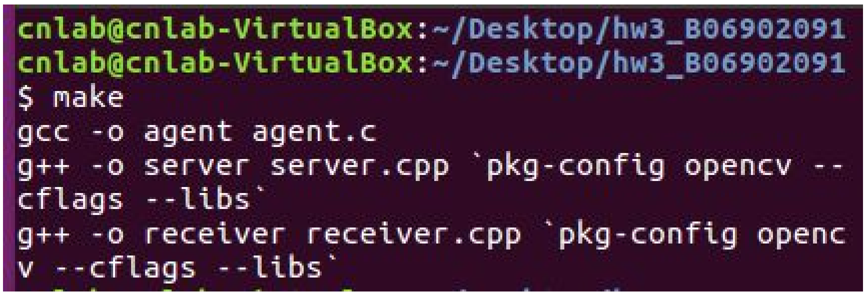
Computer network
Assignment 3

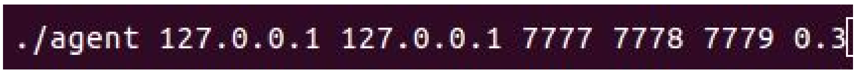
B06902091 資工四 羅寶瑩

1. **How to execute your program**
   i. Put video files (.mpg) into the folder “video”
   ii. Type “make” and there are 3 executables: server, agent and receiver.

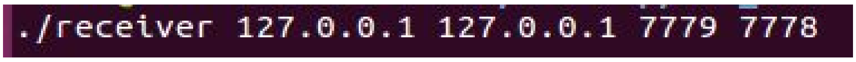


```
cnlab@cnlab-VirtualBox:~/Desktop/hw3_B06902091
cnlab@cnlab-VirtualBox:~/Desktop/hw3_B06902091
$ make
gcc -o agent agent.c
g++ -o server server.cpp `pkg-config opencv --
cflags --libs`
g++ -o receiver receiver.cpp `pkg-config openc
v --cflags --libs`
```

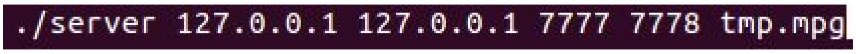
   iii. Open 3 terminals and type the following commands respectively
   - ./agent <sender IP> <recv IP> <sender port> <agent port> <recv port> <loss_rate>
   - ./receiver <receiver IP> <agent IP> <receiver port> <agent port>
   - ./server <server IP> <agent IP> <server port> <agent port> <filename>



```
./agent 127.0.0.1 127.0.0.1 7777 7778 7779 0.3
```



```
./receiver 127.0.0.1 127.0.0.1 7779 7778
```



```
./server 127.0.0.1 127.0.0.1 7777 7778 tmp.mpg
```

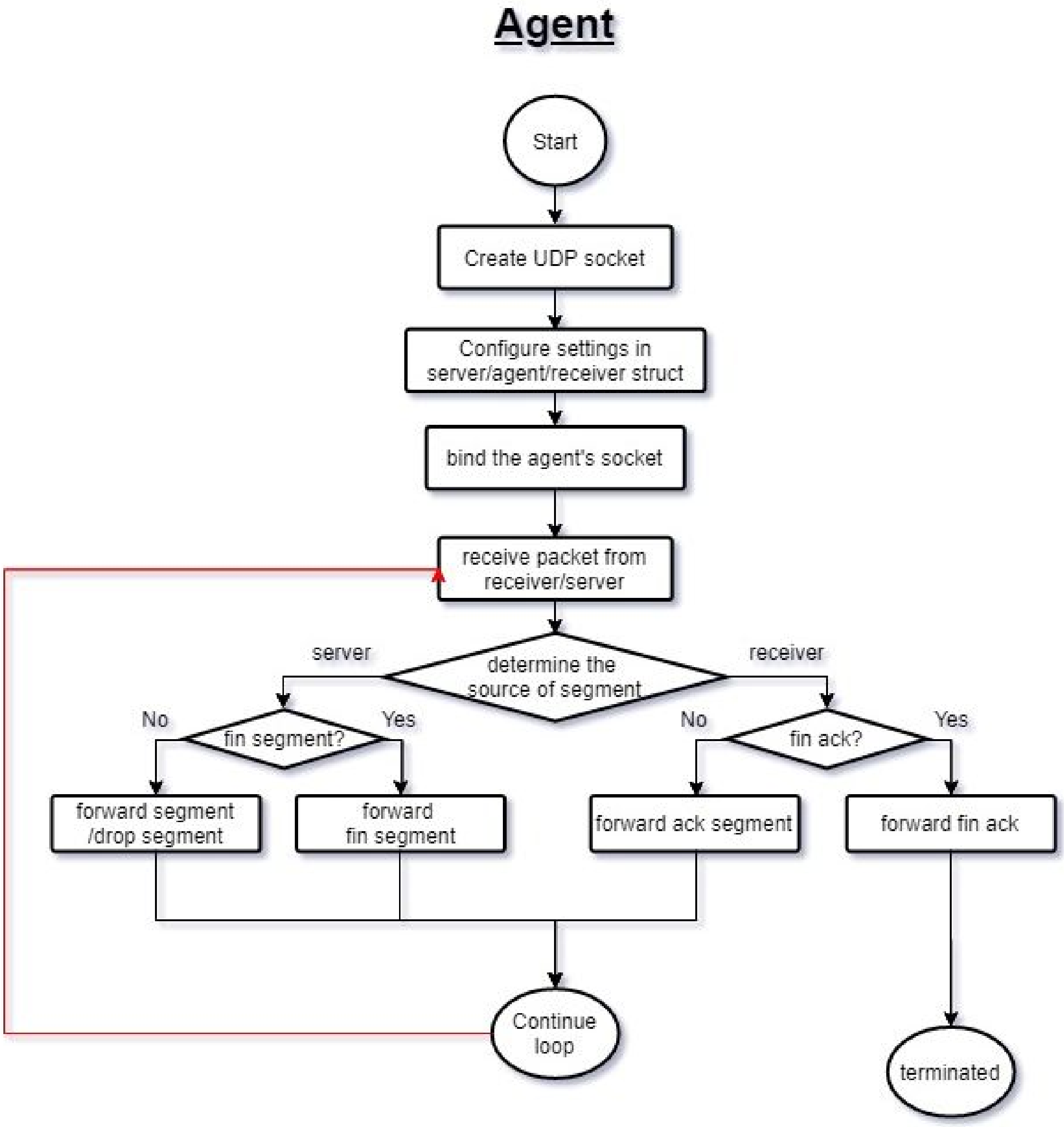
   iv. Press “Enter” in the terminal of the server if you want to terminate the server.

2. **Explain your program structure**
   a. **Agent**

First, create a UDP socket for the agent and configure settings in server, agent and receiver struct. Bind the socket of the agent.

Then, use a while loop to receive segments from server and receiver. If the segment comes from the server, calculate the drop rate to decide whether the agent would drop the segment or forward the segment to the receiver. If the segment comes from the receiver, just forward the ack segment to the server.

If the segment from the server is a fin segment, the agent would forward it to the receiver. After that, the agent would receive a fin ack segment from the receiver. Agent would forward the ack segment to the server and terminate the process.



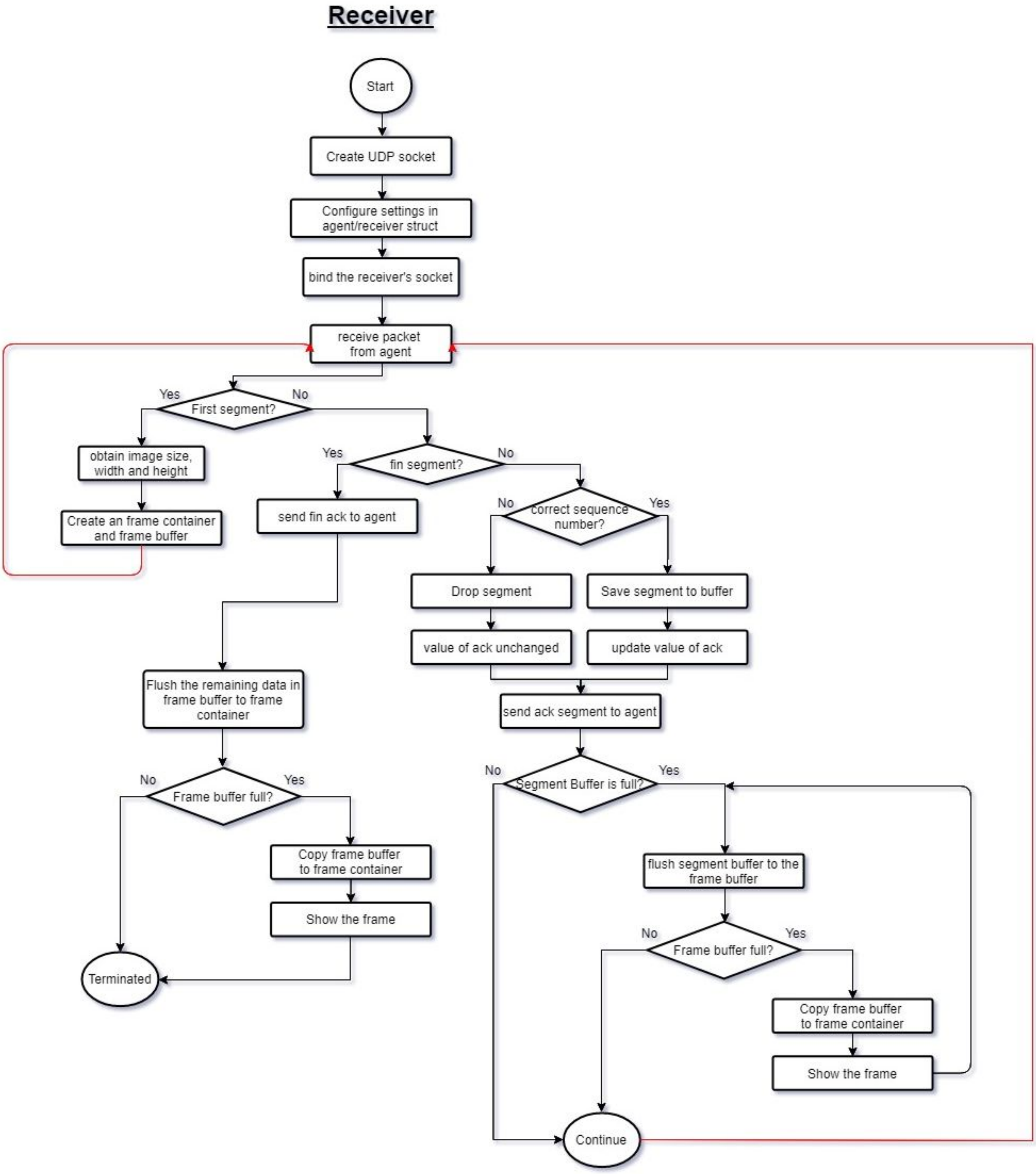
b. **Receiver**

First, create a UDP socket for the receiver and configure settings in the agent and receiver struct. Bind the socket of the receiver.

Then, use a while loop to receive segments from the agent. If the segment is the first segment, obtain the frame size, width and height. Use the information above to create a frame container and frame buffer in a suitable resolution and size. If the segment is in the correct order of sequence number, save the segment into segment buffer and update the value of the last acked segment. Otherwise, drop the segment.

After dealing with the segment, determine whether the segment buffer is full or not. If the segment buffer is full, flush the buffer to the frame buffer. If the frame buffer is completed/full, copy it into the frame container and show the frame. Reset the frame buffer and continue flushing the segment buffer.

If the receiver receives a fin segment, send the fin ack to the agent and stop the loop. Flush the data in the frame buffer to the frame container. If the frame buffer is full, copy it into the frame container and show the frame. Terminate the process.


Receiver
Start
Create UDP socket
Configure settings in agent/receiver struct
bind the receiver's socket
receive packet from agent
Yes
First segment?
No
obtain image size, width and height
Create an frame container and frame buffer
Yes
fin segment?
No
send fin ack to agent
No
correct sequence number?
Yes
Drop segment
Save segment to buffer
value of ack unchanged
update value of ack
send ack segment to agent
Flush the remaining data in frame buffer to frame container
No
Frame buffer full?
Yes
Copy frame buffer to frame container
Show the frame
Terminated
No
Segment Buffer is full?
Yes
flush segment buffer to the frame buffer
No
Frame buffer full?
Yes
Copy frame buffer to frame container
Show the frame
Continue

c. **Server**

First, create a UDP socket for the server and configure settings in the agent and server struct. Bind the socket of the server. Capture the frames of the video and get the width and height of the frames. Create a frame container to store the frame of the video.

Then, use a while loop to get the frame of the video and calculate the size of the frame. For the first segment, we send the frame size, width, and height to the agent. After that, we use a for loop to repeat the following steps:

1. copy the data in the frame to the segment
2. send the segment to the agent.
3. If the number of segment sent >= window size, stop the for loop

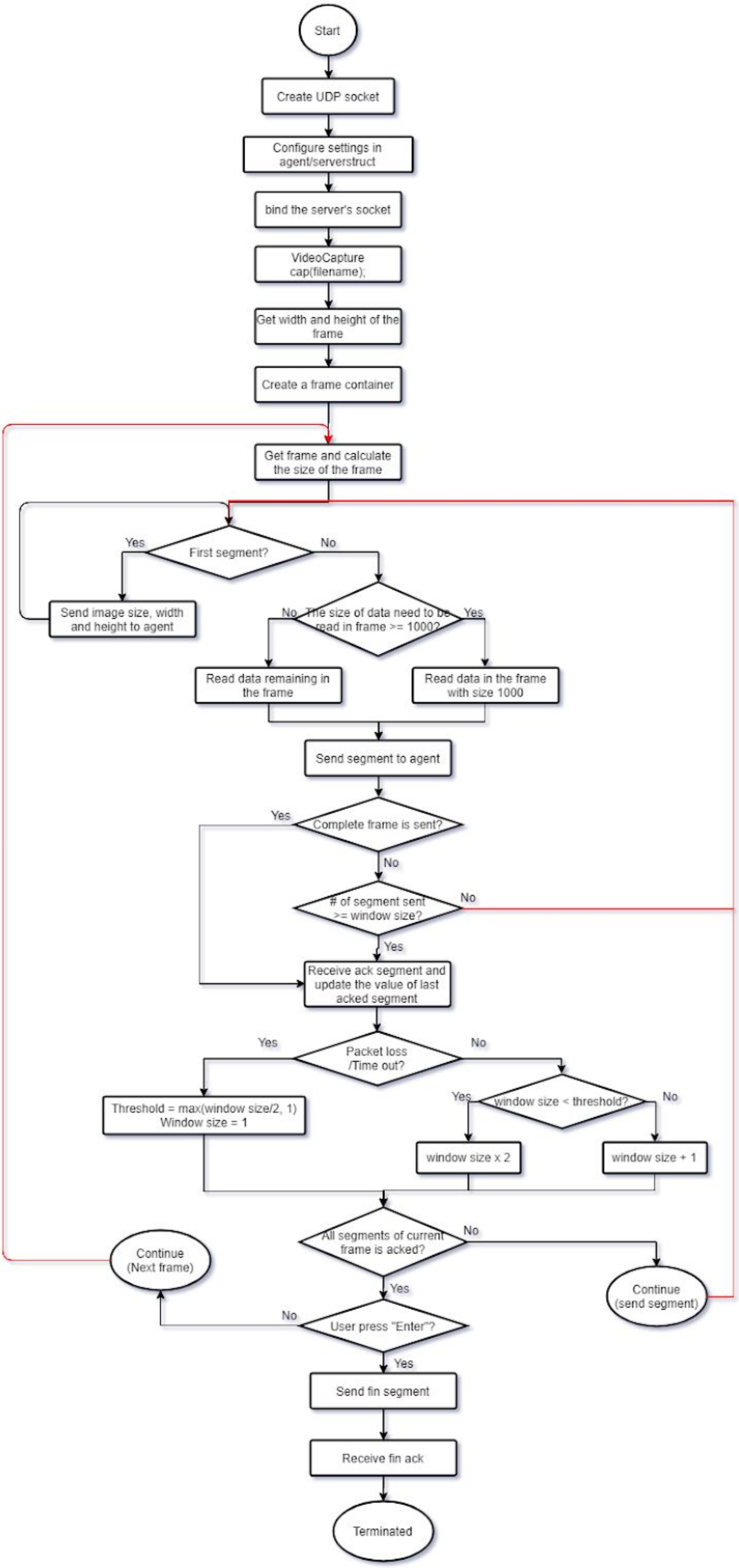
After sending the segment to the agent, we use a loop to receive the ack segment sent by the agent(also receiver) and update the value of the last acked segment.

If there are packet loss or timeout events, set the threshold to max((window size/2),1), set window size to 1 and retransmit the segments. Otherwise, if window size < threshold, double the window size. If window size >= threshold, add 1 to window size.

When all segments of the current frame is acked, repeat the steps above for the next frame of video.

If the user wants to stop the server or all of the frame of video is transferred, send a fin segment to the agent. After receiving the fin ack from the agent, the process would be terminated.

Server


Start
Create UDP socket
Configure settings in agent/serverstruct
bind the server's socket
VideoCapture cap(filename);
Get width and height of the frame
Create a frame container
Get frame and calculate the size of the frame
Yes
First segment?
No
Send image size, width and height to agent
No
The size of data need to be read in frame >= 1000?
Yes
Read data remaining in the frame
Read data in the frame with size 1000
Send segment to agent
Yes
Complete frame is sent?
No
# of segment sent >= window size?
No
Yes
Receive ack segment and update the value of last acked segment
Yes
Packet loss /Time out?
No
Threshold = max(window size/2, 1) Window size = 1
Yes
window size < threshold?
No
window size x 2
window size + 1
All segments of current frame is acked?
No
Continue (Next frame)
Yes
Continue (send segment)
No
User press "Enter"?
Yes
Send fin segment
Receive fin ack
Terminated

Reference:

1. ***Setting timeout of recv***
   https://stackoverflow.com/questions/30395258/setting-timeout-to-recv-function/30395738
2. ***exit loop by key pressing***
   https://web.archive.org/web/20180401093525/http://cc.byexamples.com/2007/04/08/non-blocking-user-input-in-loop-without-ncurses/
3. ***end of video capture***
   https://stackoverflow.com/questions/21235885/end-of-video-file-opencv