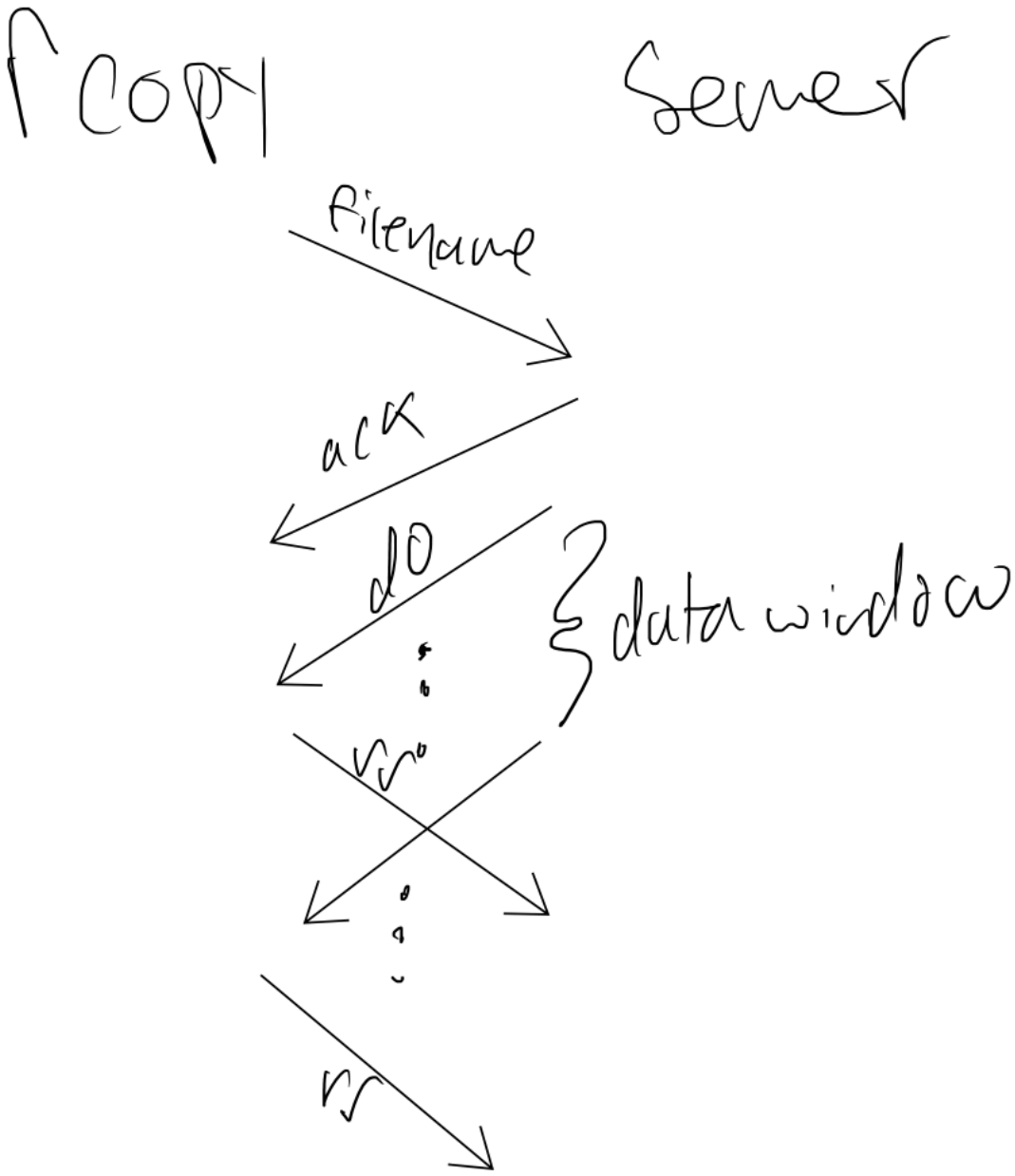
Tab 1

Cooper Mattern
Armaan Oberai

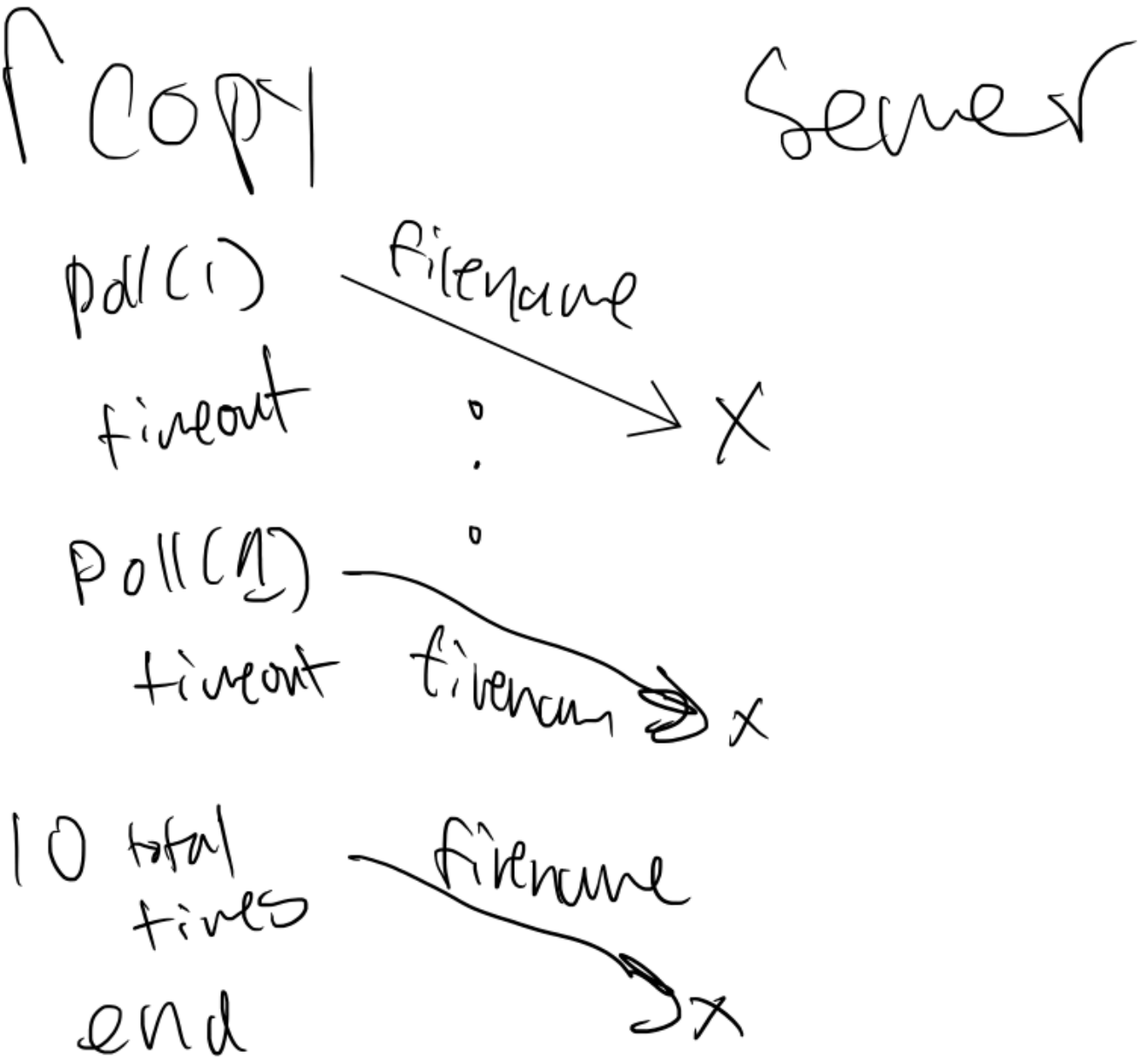
Program 3 - SREJ Design Assignment

PART I - DESIGN

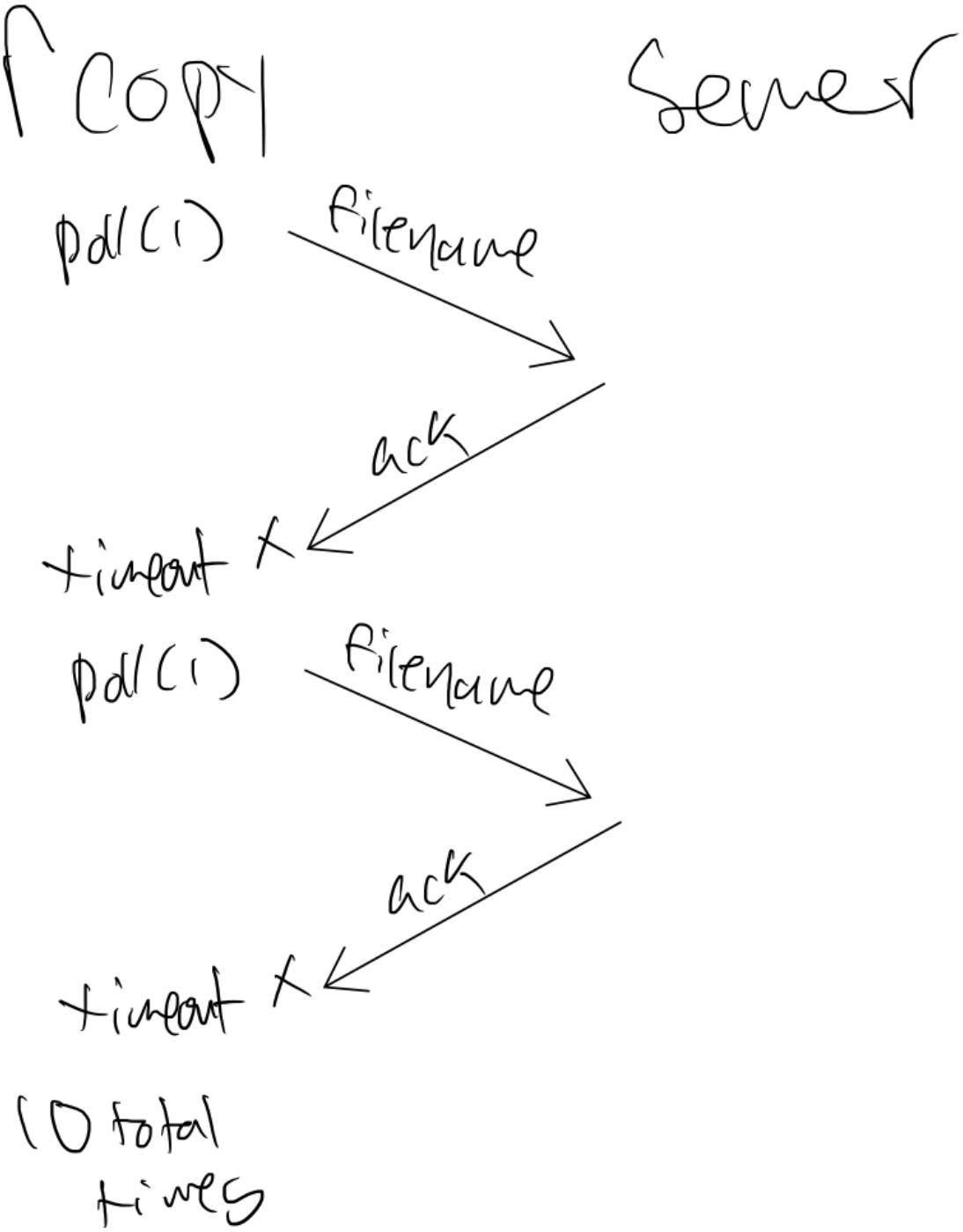
3a. No packets lost



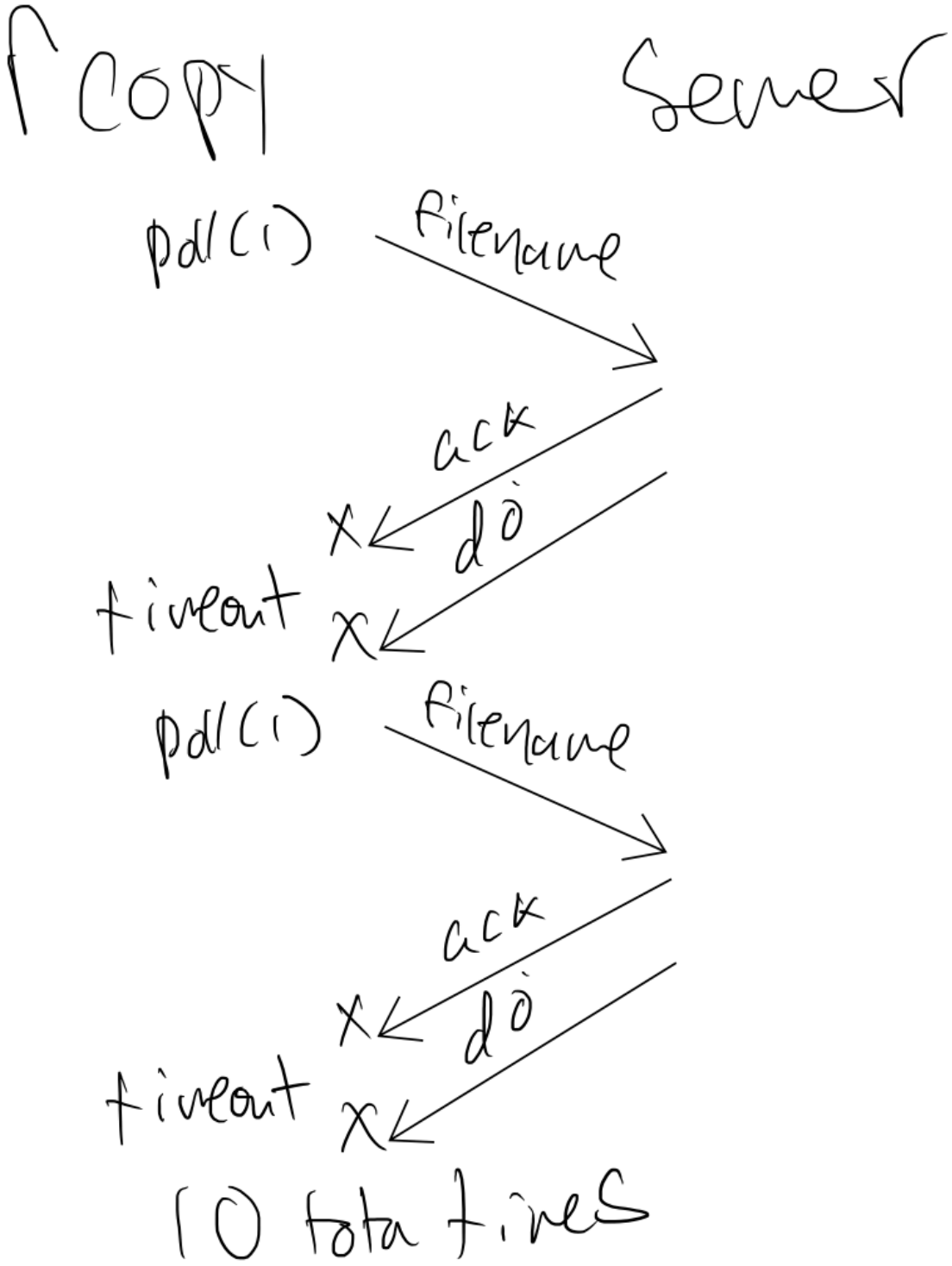
3b. Filename transmission from rcopy gets lost everytime



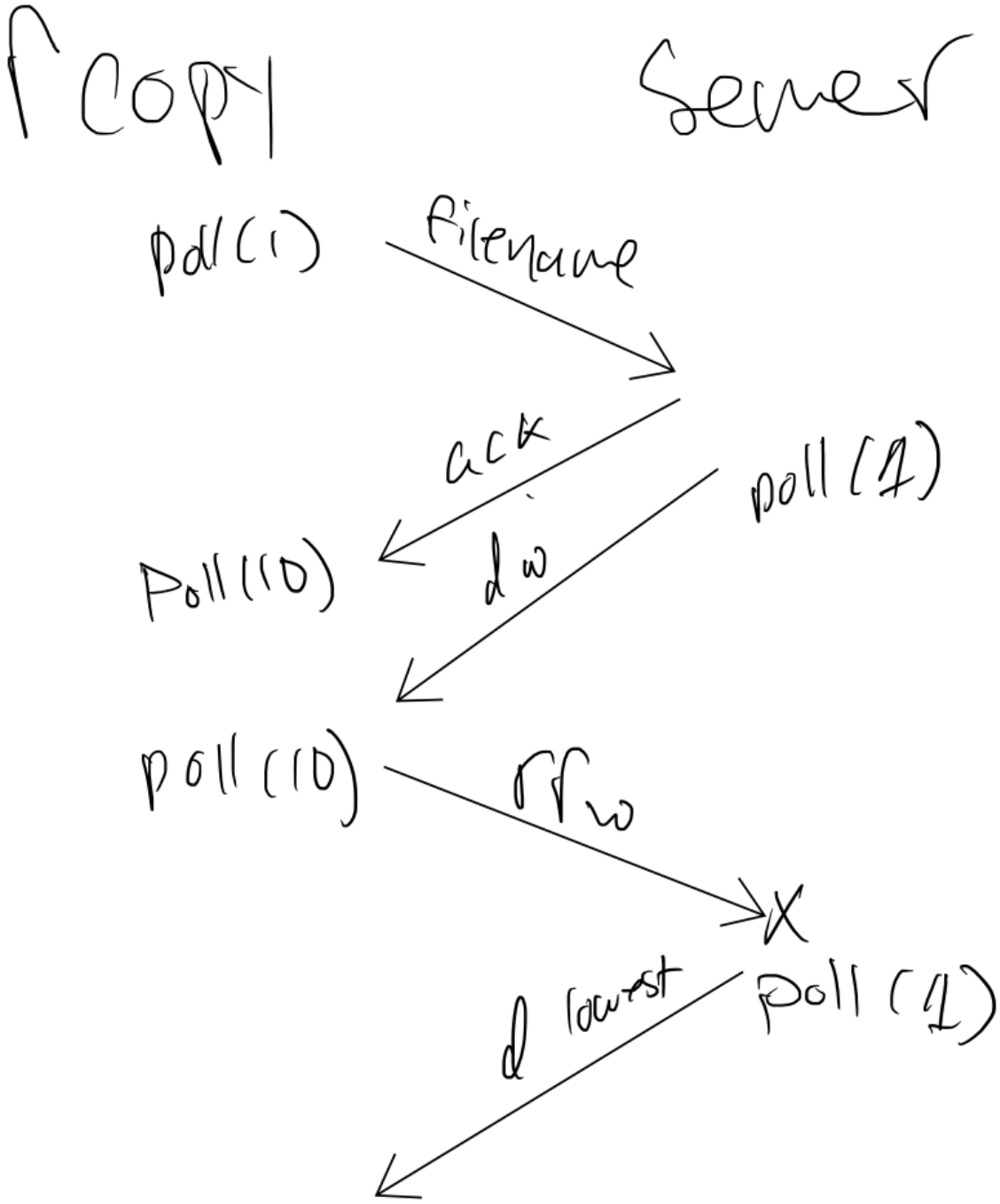
3c. Filename ack from server gets lost everytime



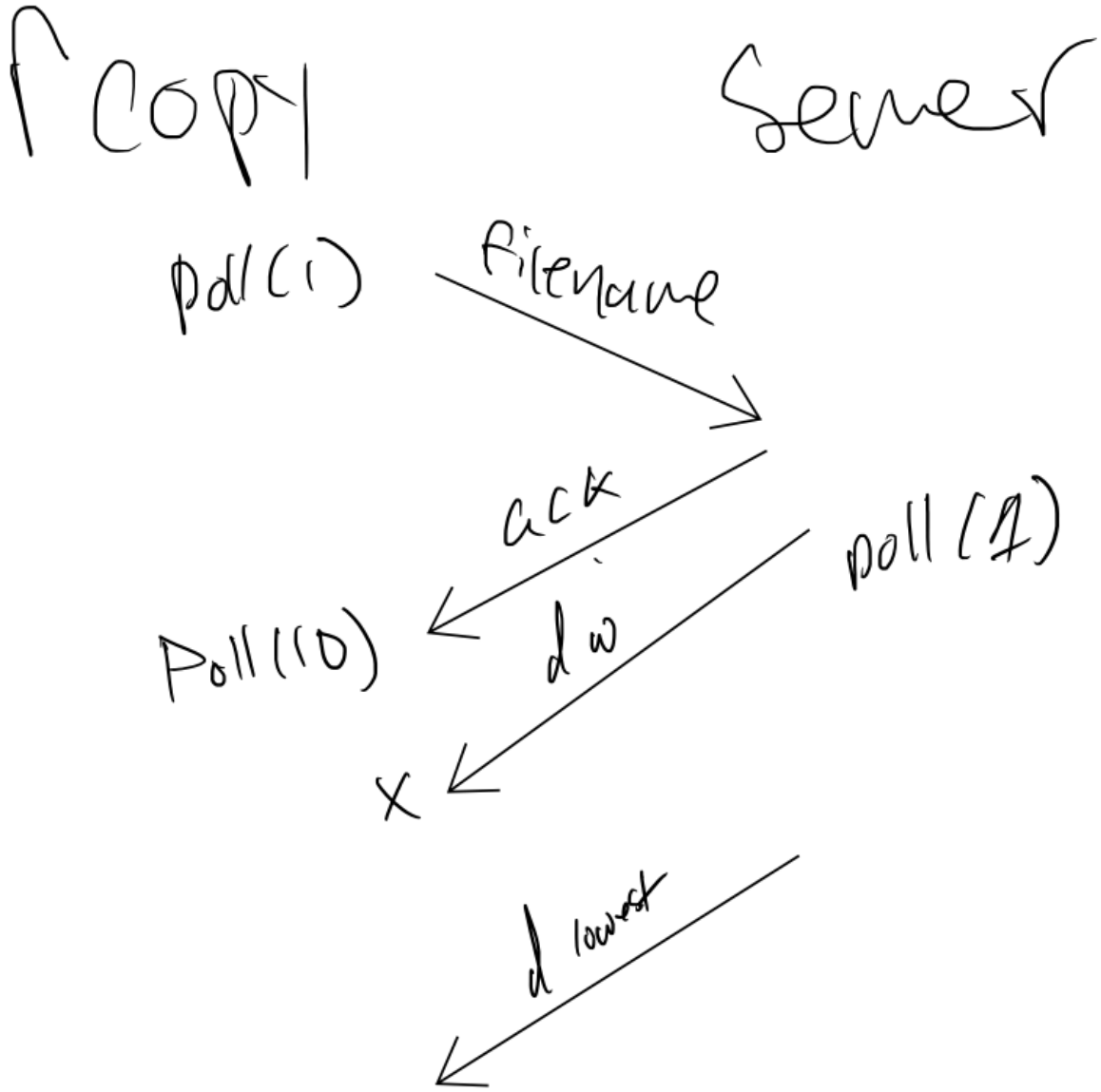
3d. Filename ack and data packet from server is lost



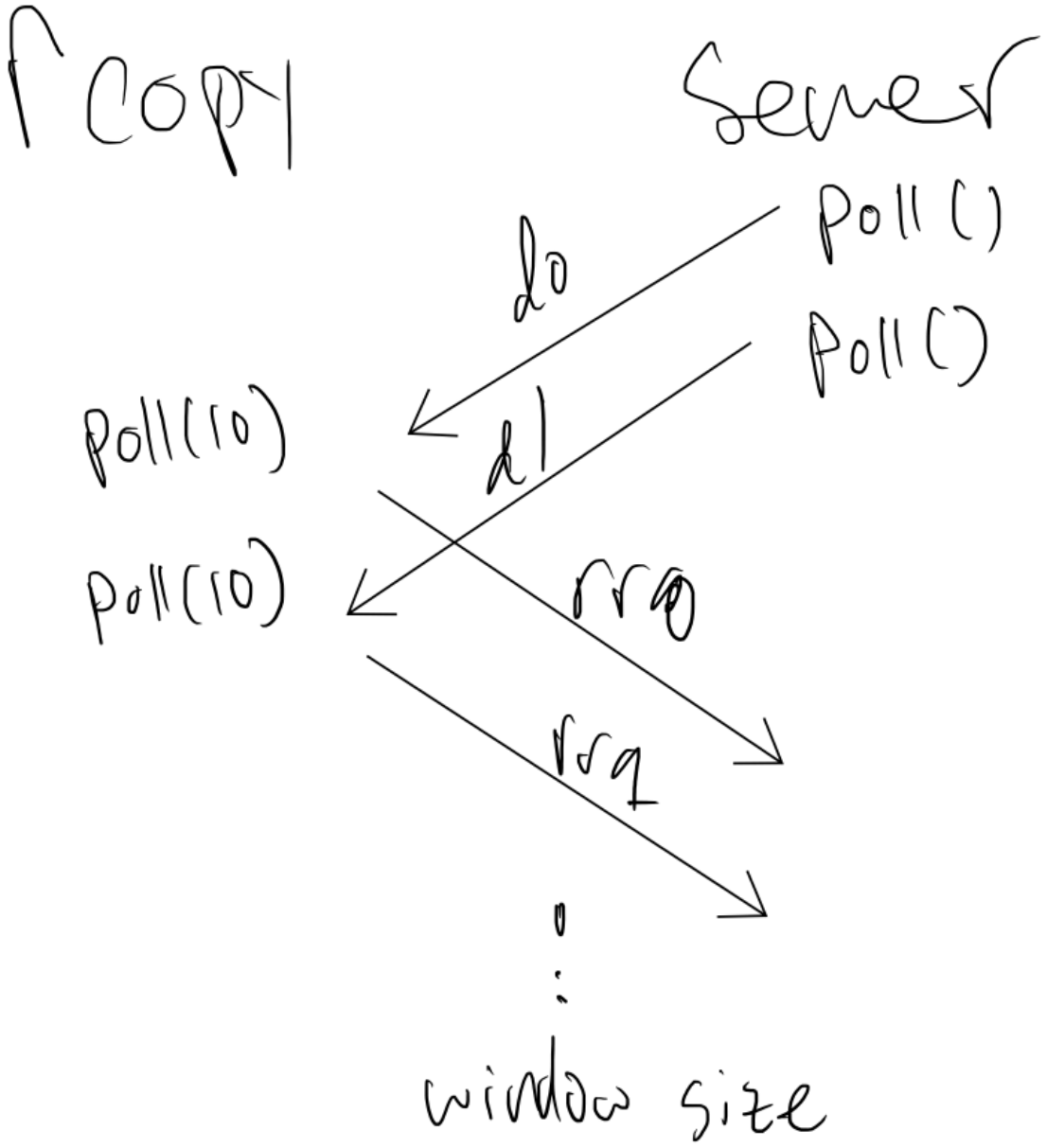
3e. First data window is lost from server



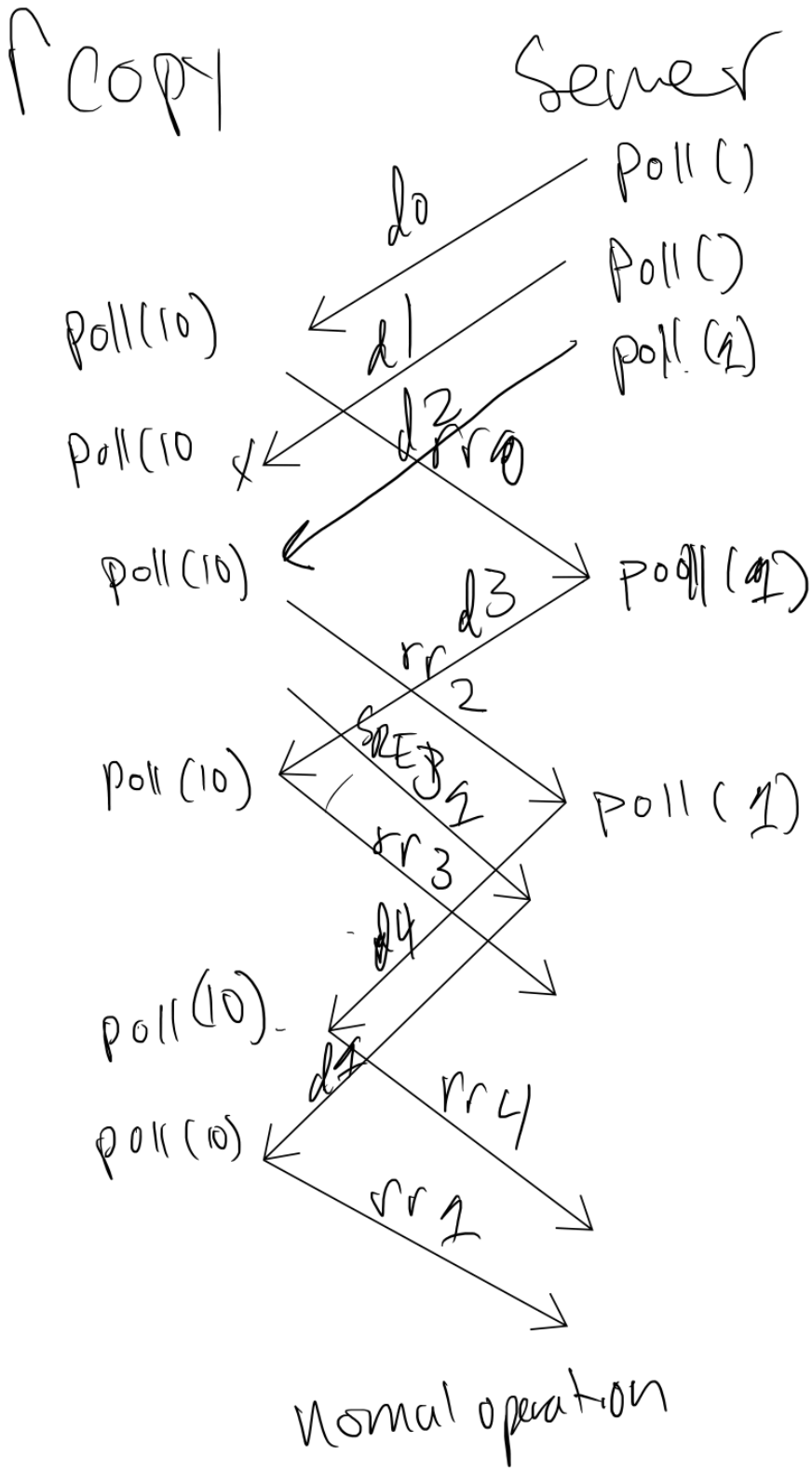
3f. First RR window from rcopy is lost



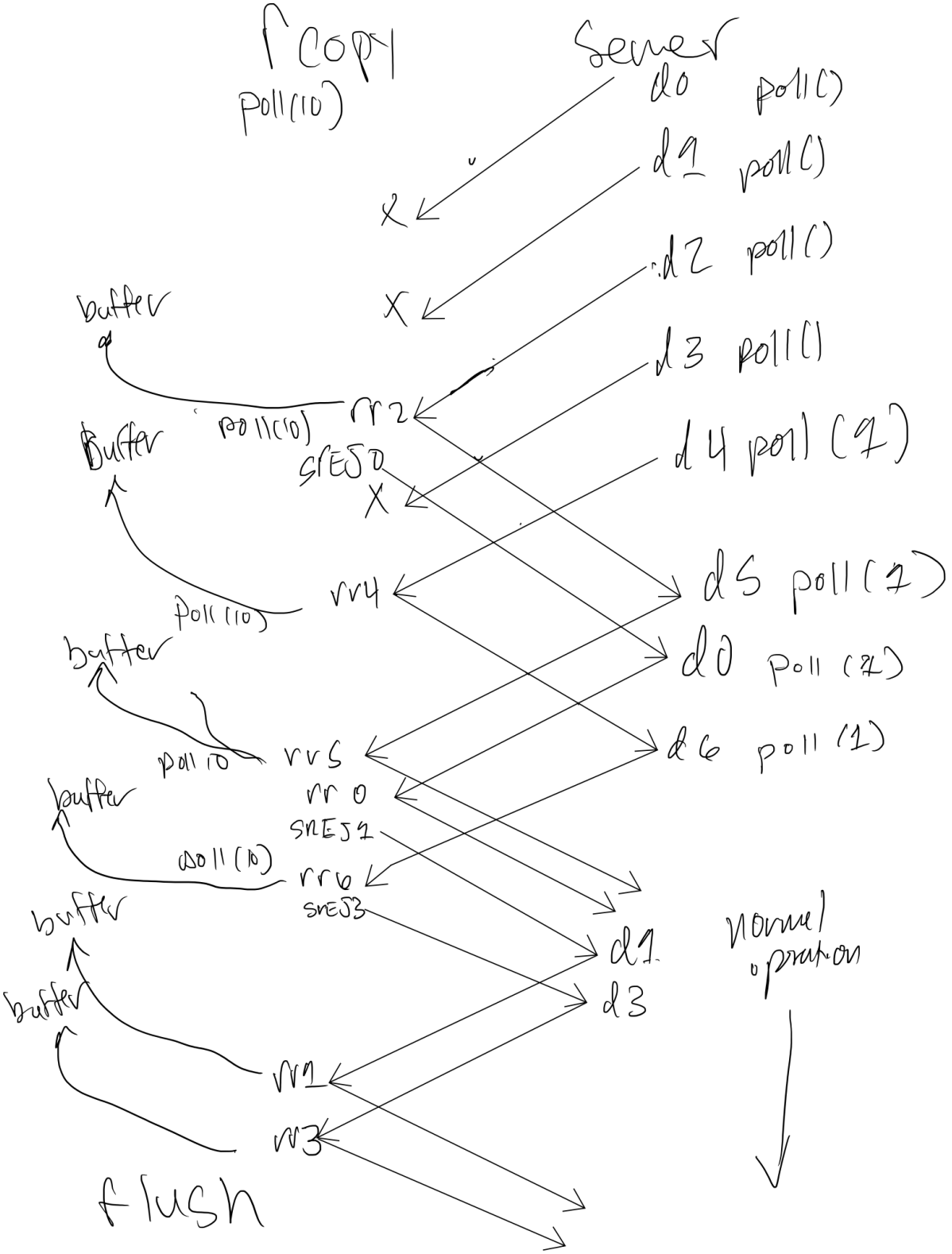
4a. Normal data window flow - no data or RRs lost



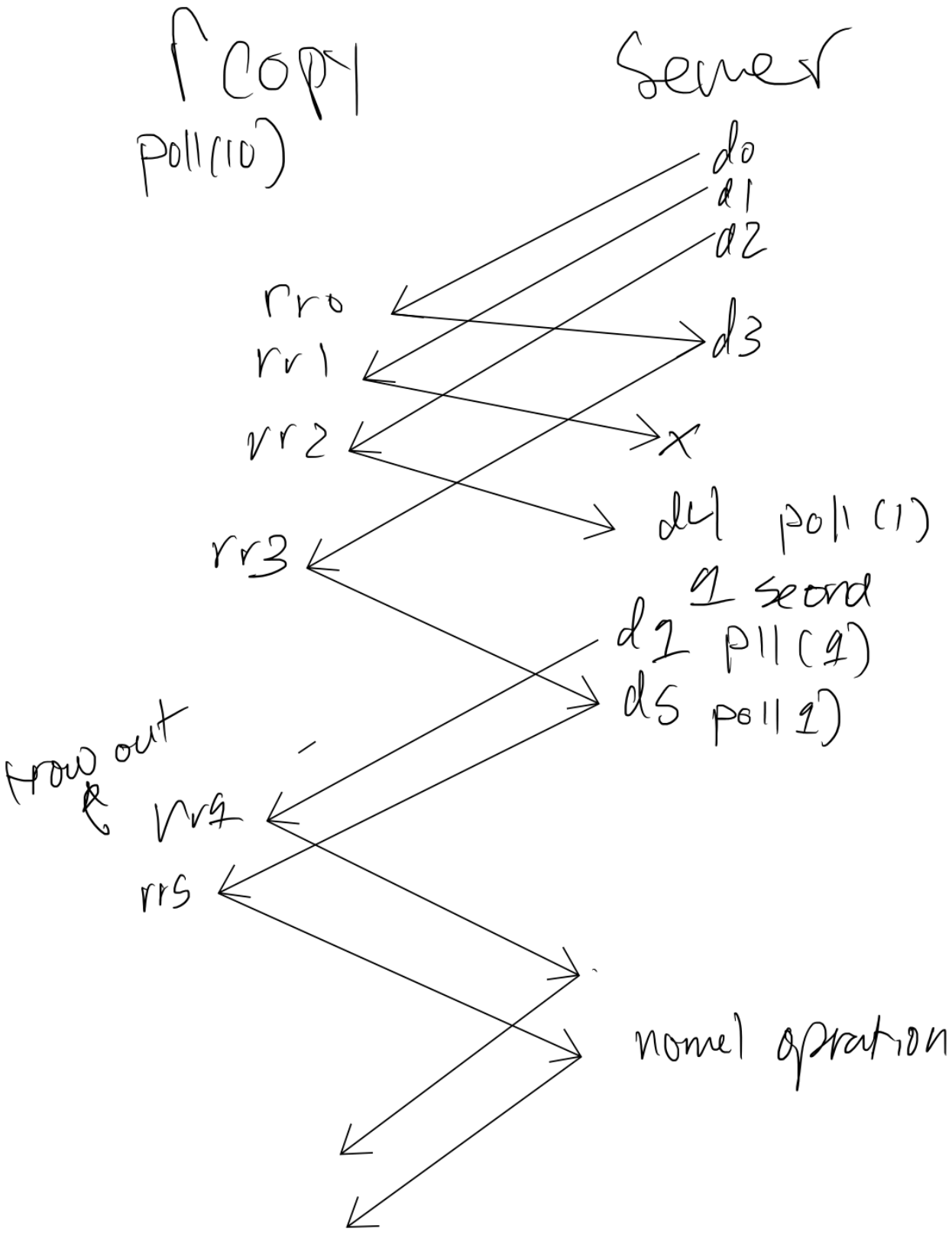
4b. One data packet lost in win_size = 3



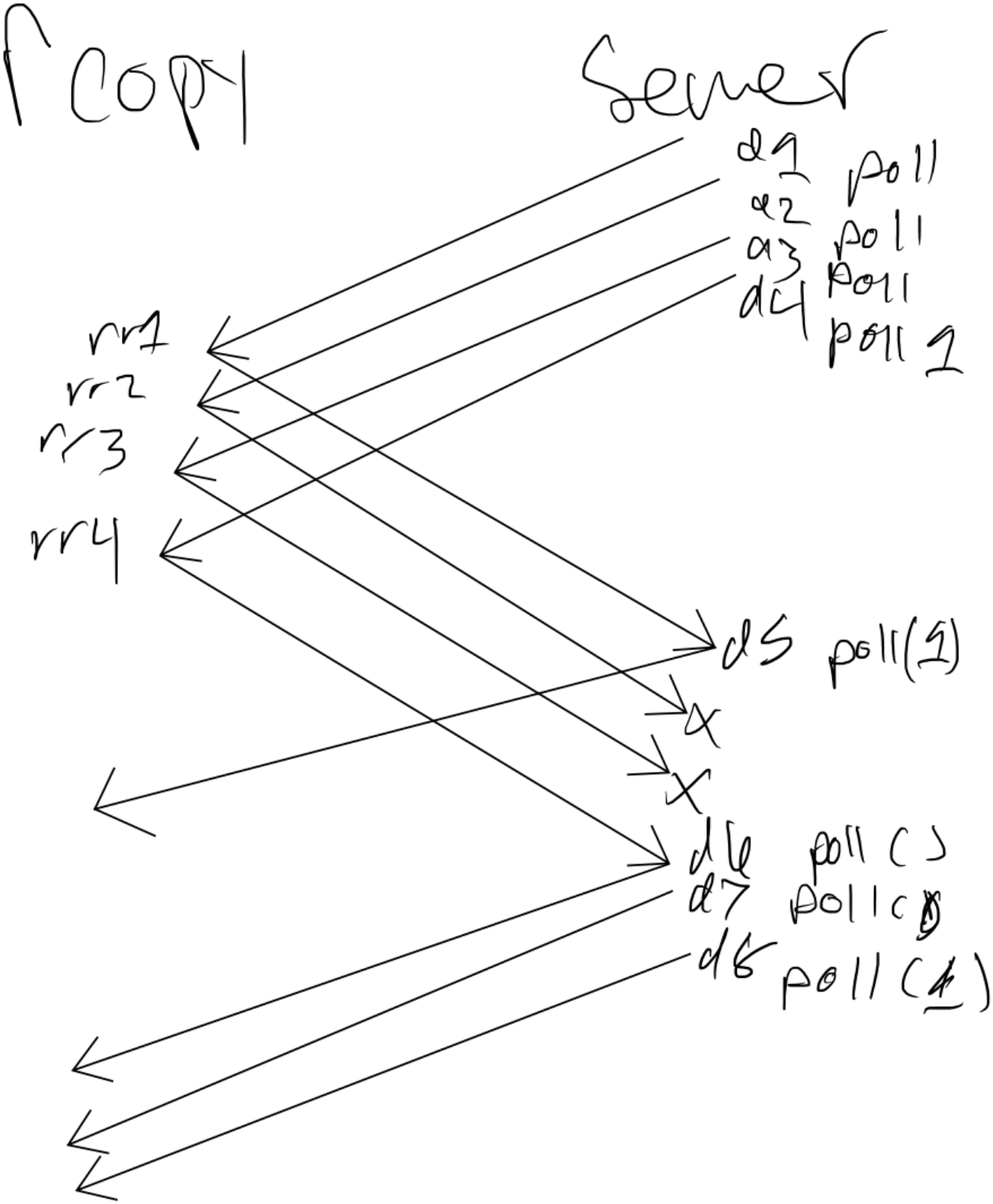
4c. Sequential and non-sequential data packets get lost in a window



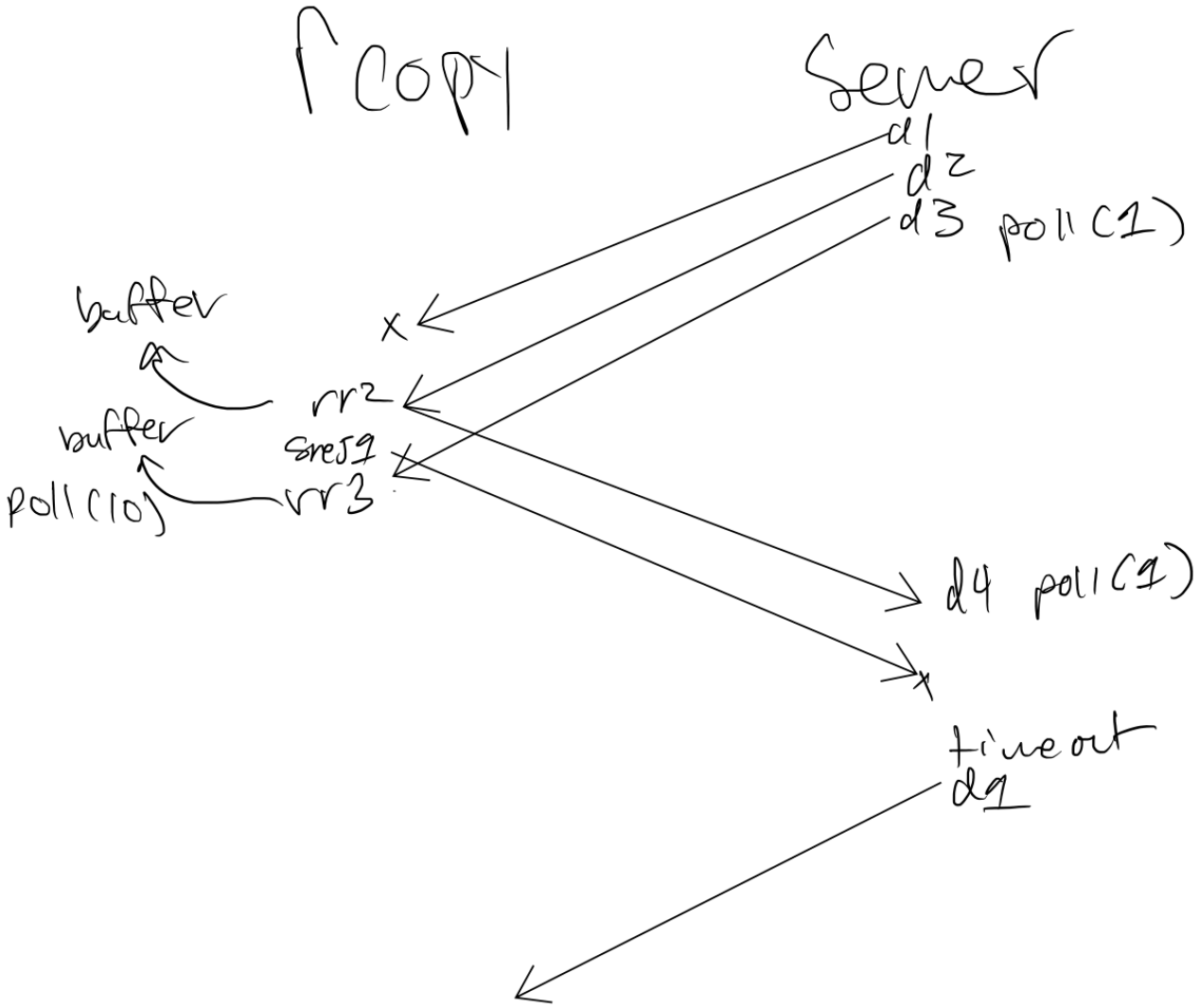
4d. RR lost for data packet 2



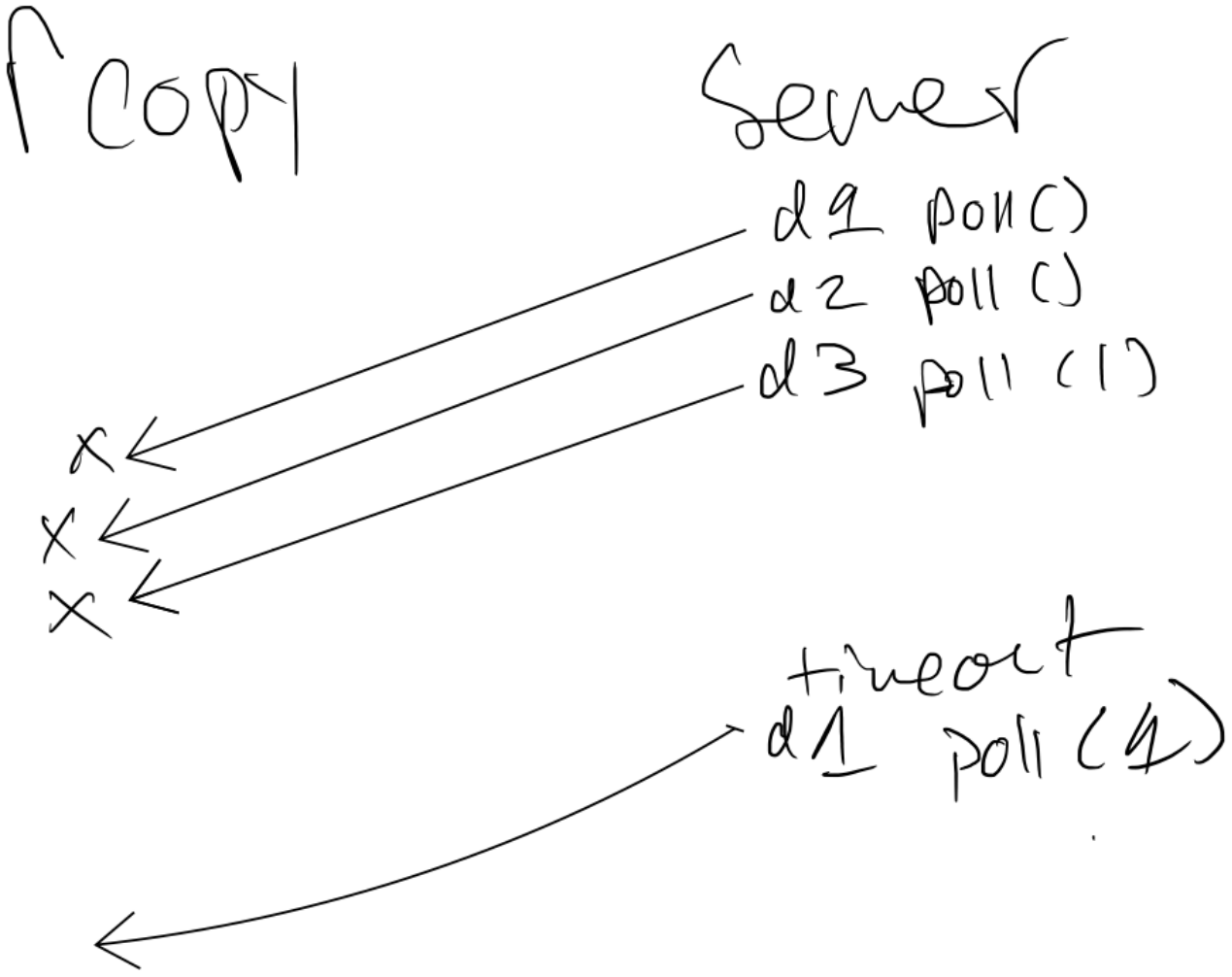
4e. Multiple RR's lost (RR2 and RR3)



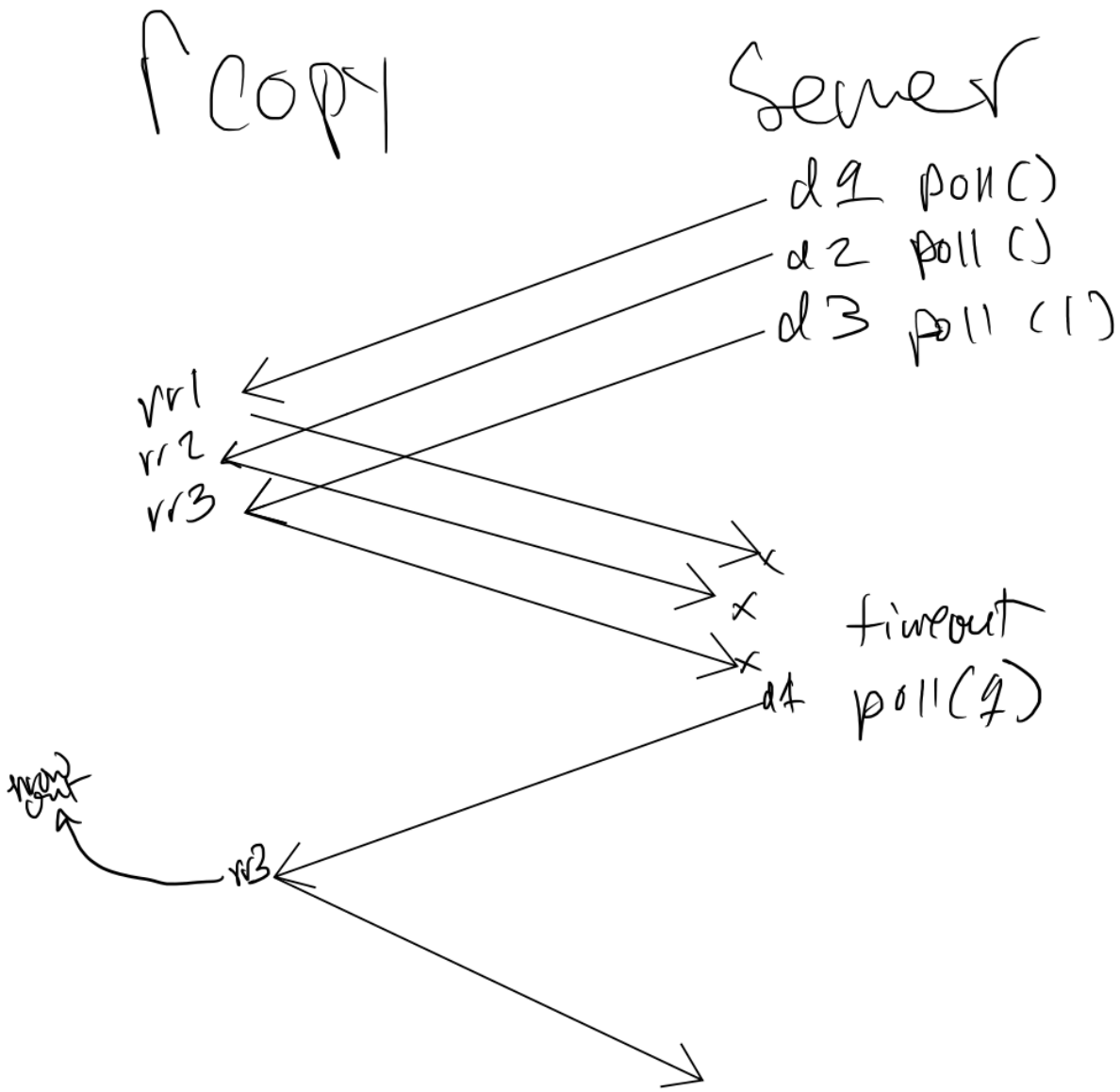
4f. SREJ lost


Copy
Server
d1
d2
d3
poll(1)
buffer
x
buffer
rr2
srej1
poll(10)
rr3
d4 poll(1)
timeout
d1

4g. Entire window of data is lost



4h. Entire window or RRs is lost



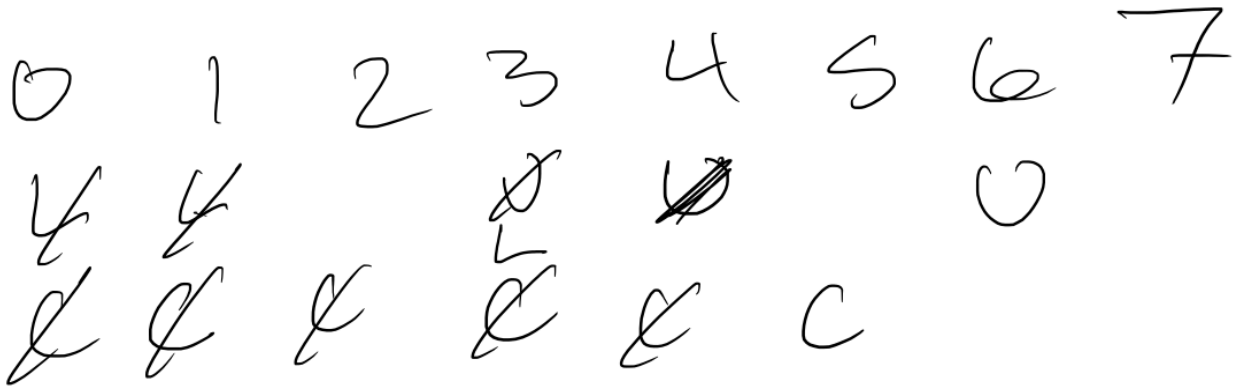
5. The scenarios presented in parts (f), (g), and (h) always necessarily lead to the window closing and leading to a timeout to send the lowest un-ACK'd packet. If the recovered data packet from a SREJ is lost, that should also lead to the window being closed and timing out. For scenario (d) and (e), the window will stay open as long as a higher RR gets sent to the server.

6a. If duplicate data is received, simply reply to it by sending the the lowest un-ACK'd RR (e.g. if data packet #2 is received but we are ready for data packet #6 → reply with RR6)

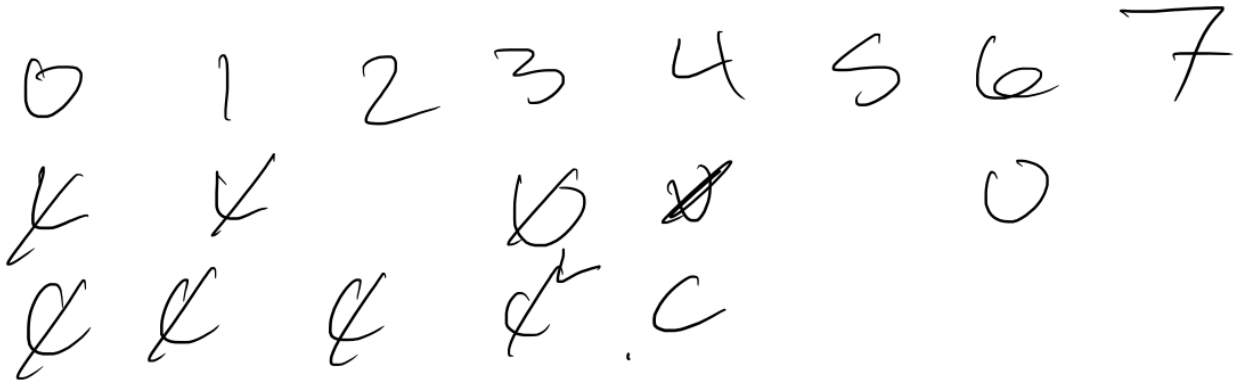
6b. Put the data in a circular buffer and reply to the server by sending RR_expected

6c. Write the payload to the disk of the expected data frame packet, then flush the buffer and reply to the server for the next lowest RR missing based on the contents of the buffer

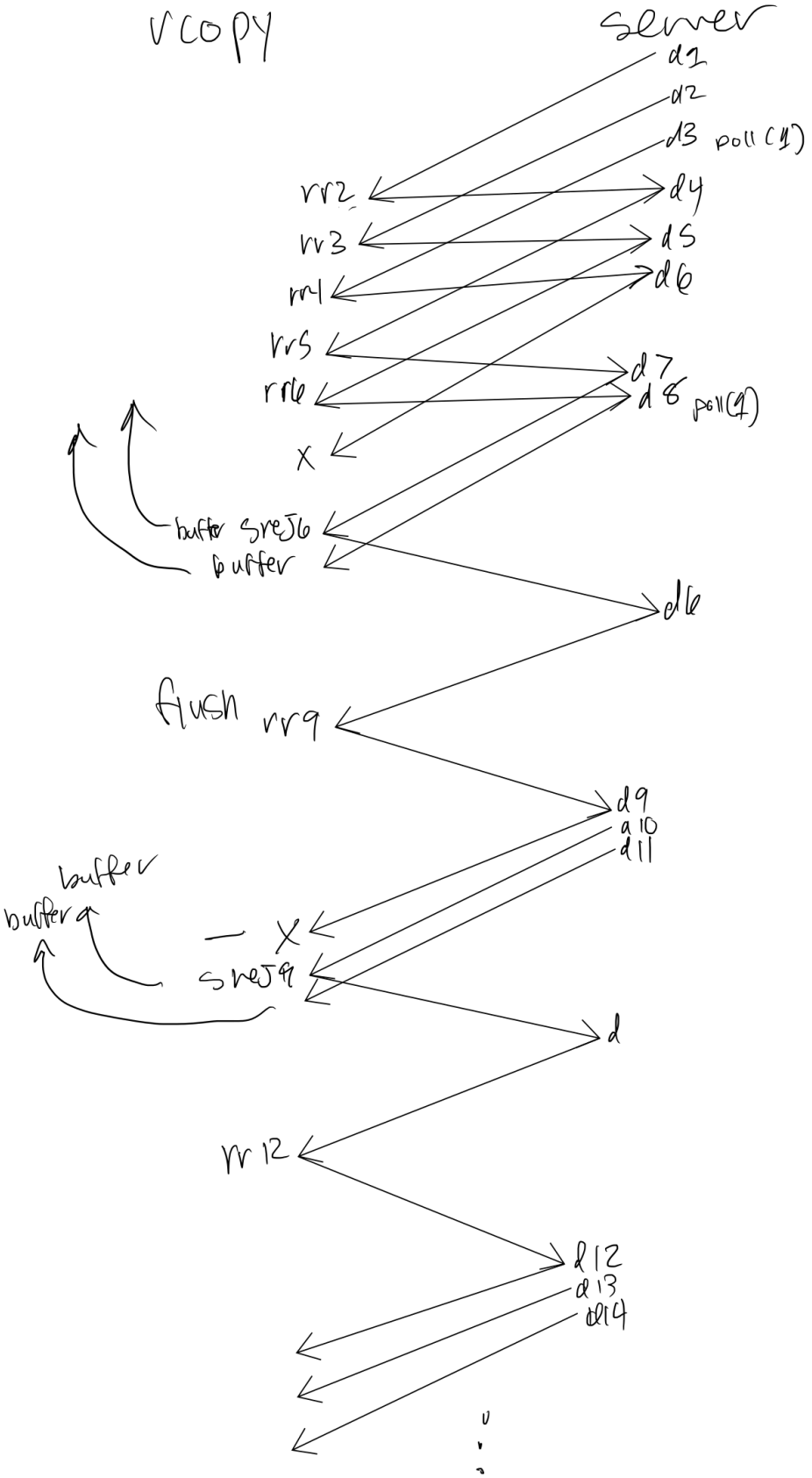
7a. Sliding window with win_size = 3 w/o SREJ



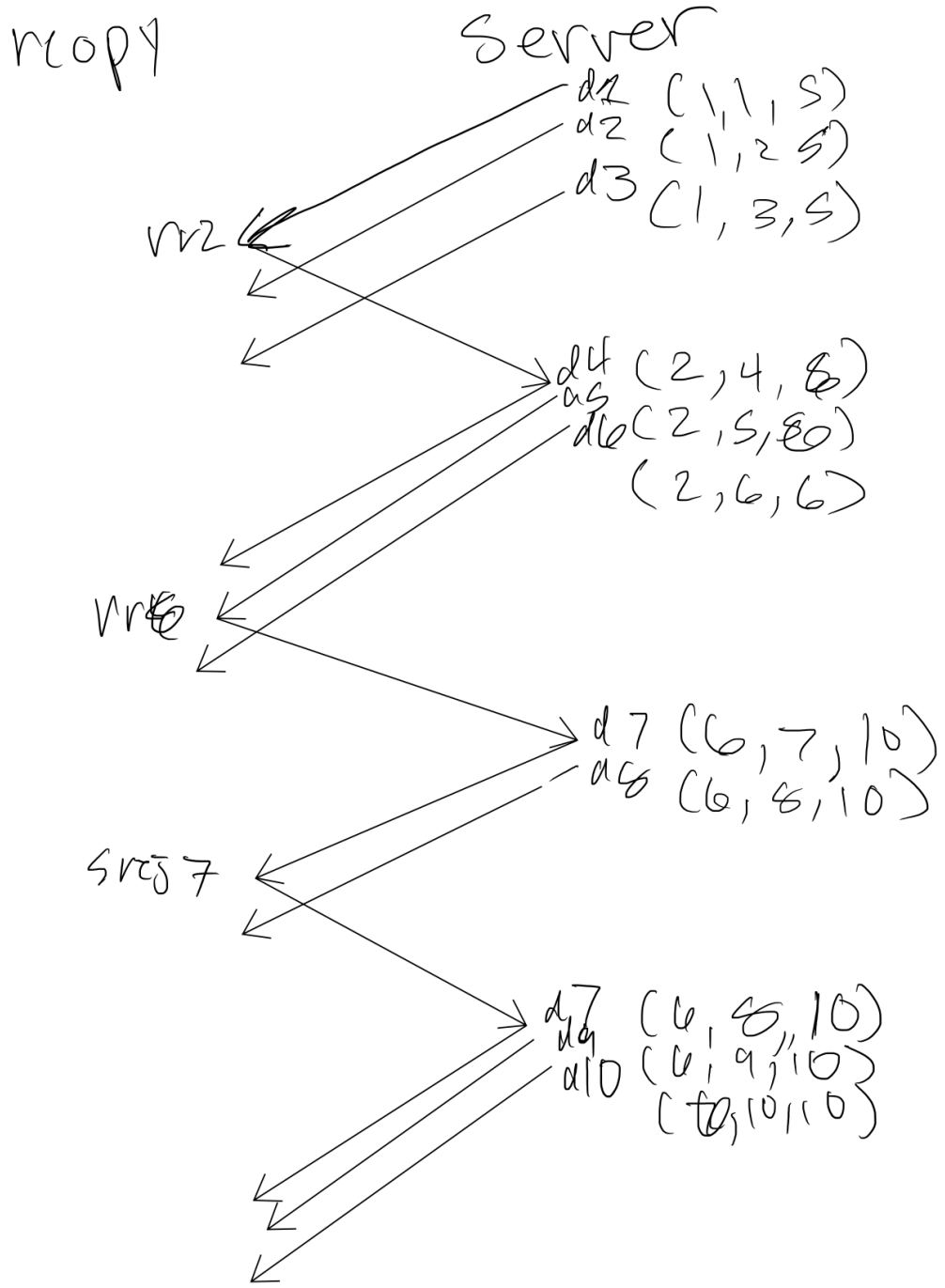
7b. Sliding window with win_size = 3 with SREJ



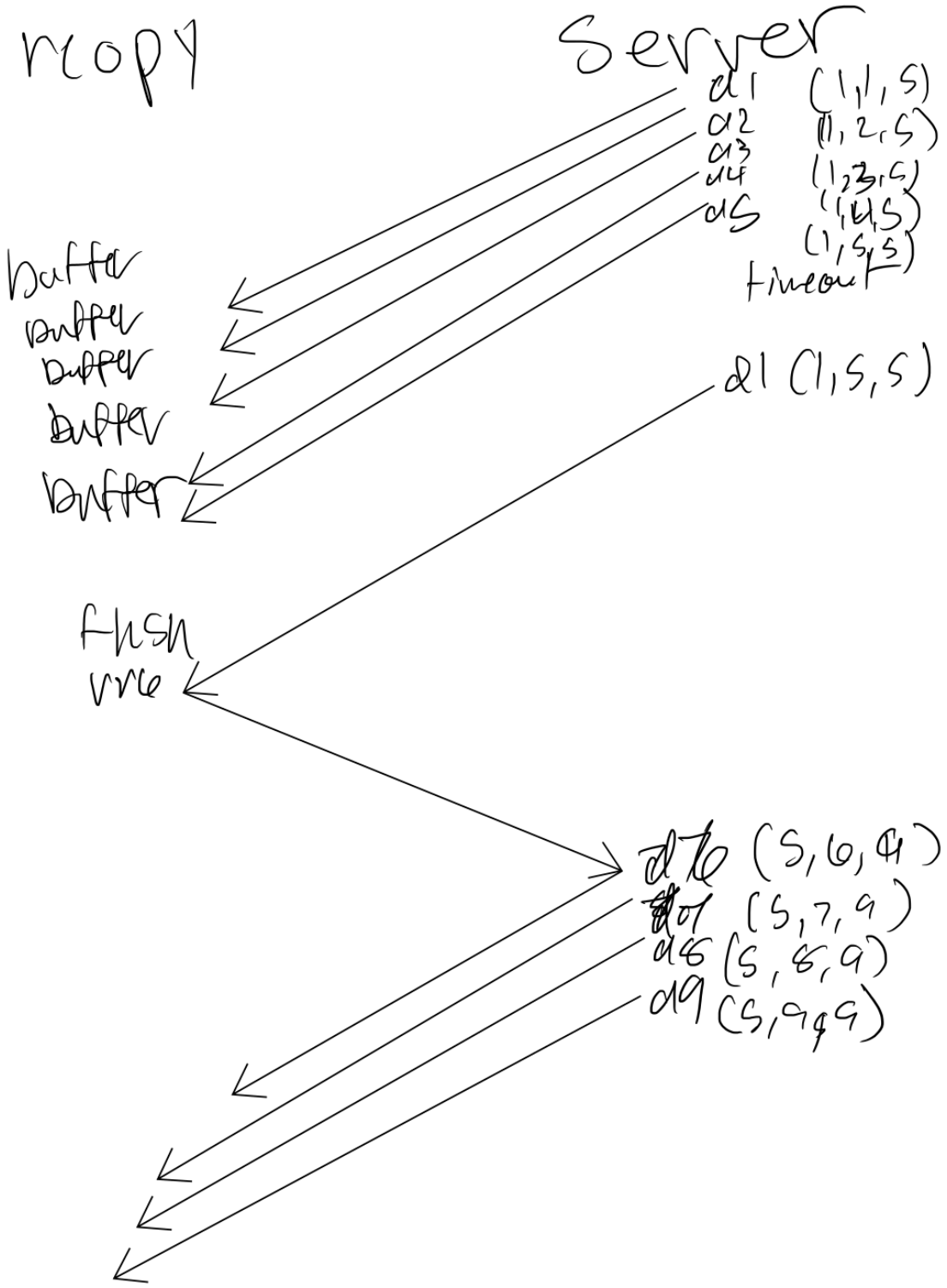
8. The SREJs are sent as soon as the client notices an out of order data packet and will send an SREJ and RR for that sequence number. Any other data packet the client sees it will simply buffer it for later. Once the requested data packet is retransmitted and received, the client will write this to the disk with the buffer and now expect a data packet from the next lowest sequence number not seen in the buffer. The only packets that are resent are those after a timeout or a data packet after an explicit SREJ request.


vcopy
server
d1
d2
d3 poll(1)
rr2
d4
rr3
d5
rr4
d6
rr5
d7
rr6
d8 poll(4)
X
buffer snd6
buffer
d6
flush rr9
d9
d10
d11
buffer
buffer
X
snd9
d
rr12
d12
d13
d14

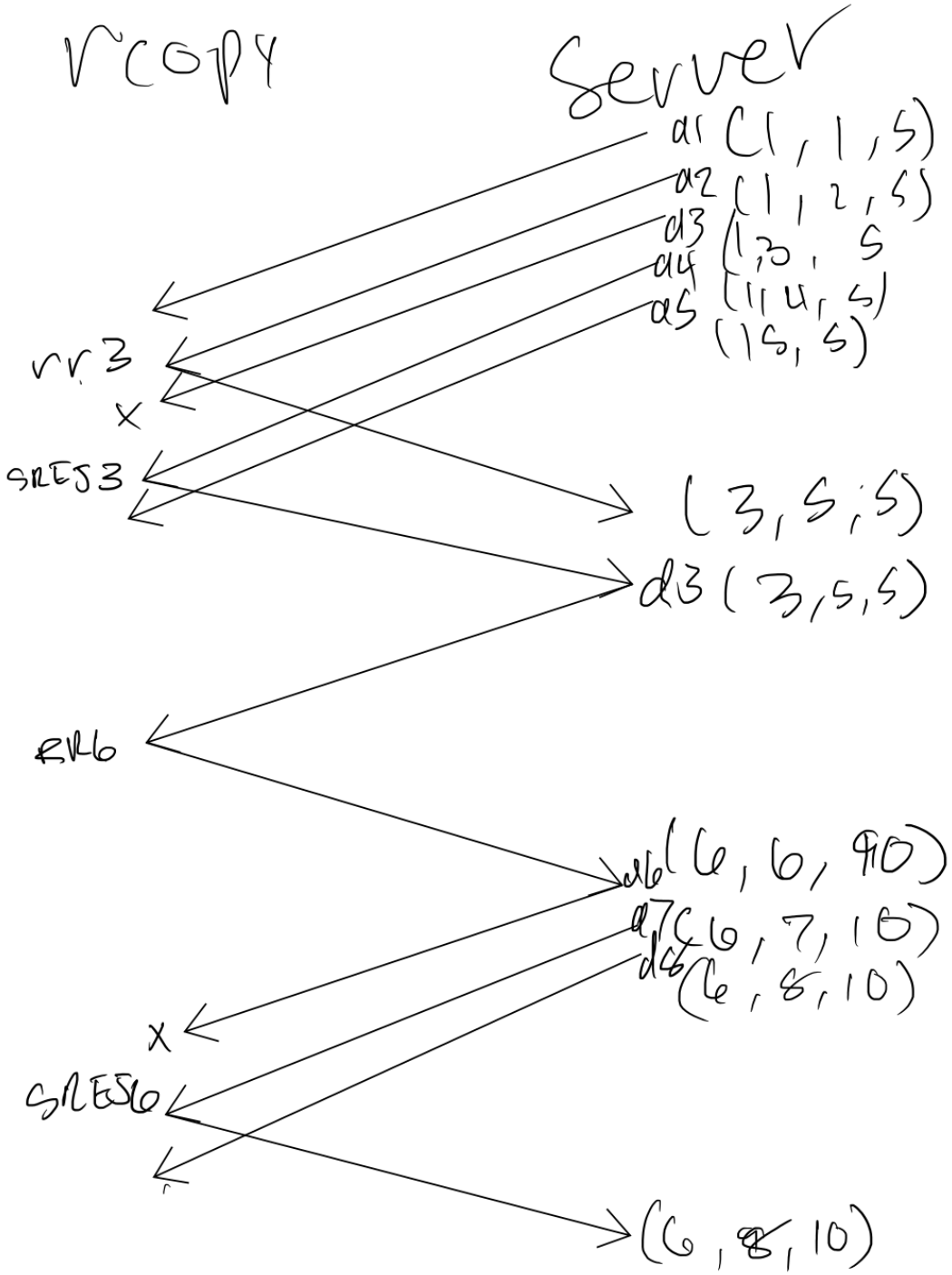
9a. With a window size of 5: sender sends data packets 1-3, sender receives RR 2, sender sends packets 4,5,6, sender receives RR 6, sender sends packets 7 and 8, sender receives SREJ 7, sender sends 7, 9 and 10



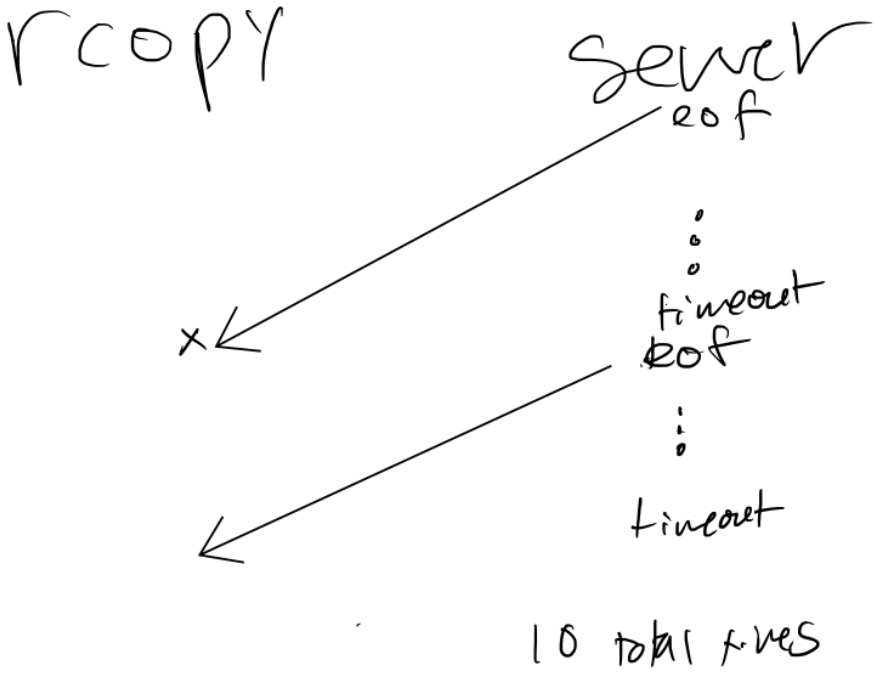
9b. With a window size of 5: sender sends data packets 1-5, sender receives nothing and timeouts, sender resends packet 1, sender receives RR 6, sender sends 6, 7, 8, 9



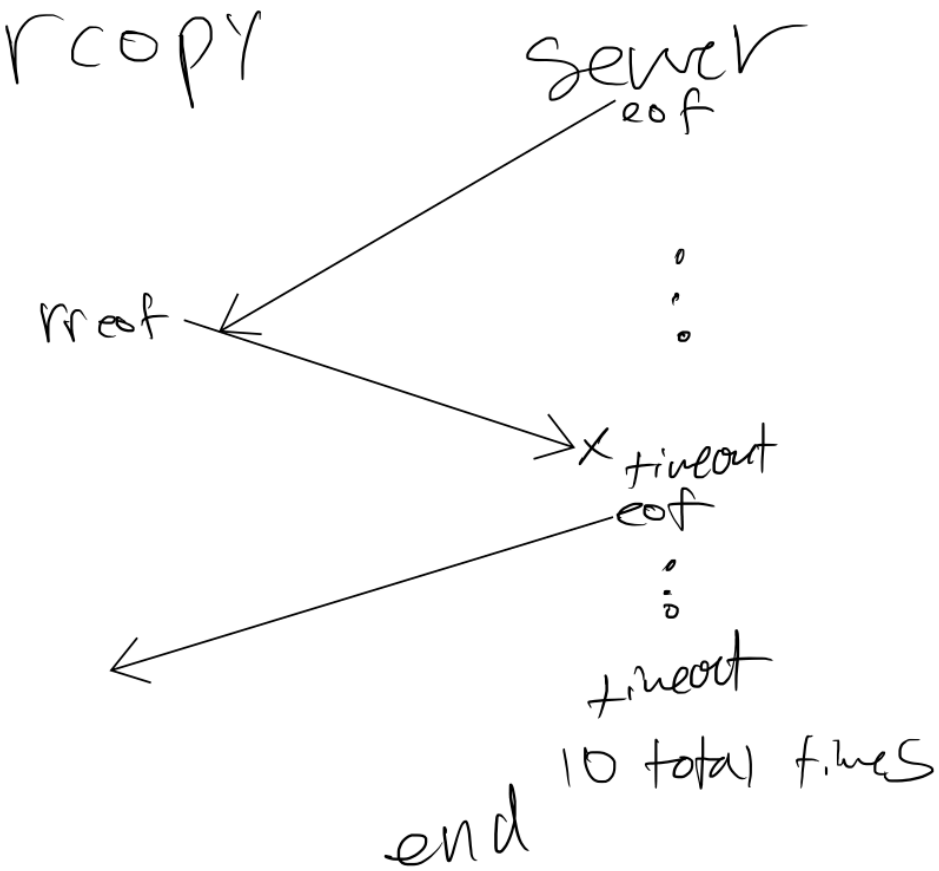
9c. With a window size of 5: sender sends data packets 1-5, sender receives RR 3, sender receives SREJ 3, sender resends packet 3, sender receives RR 6, sender sends 6, 7, 8, sender receives SREJ 6



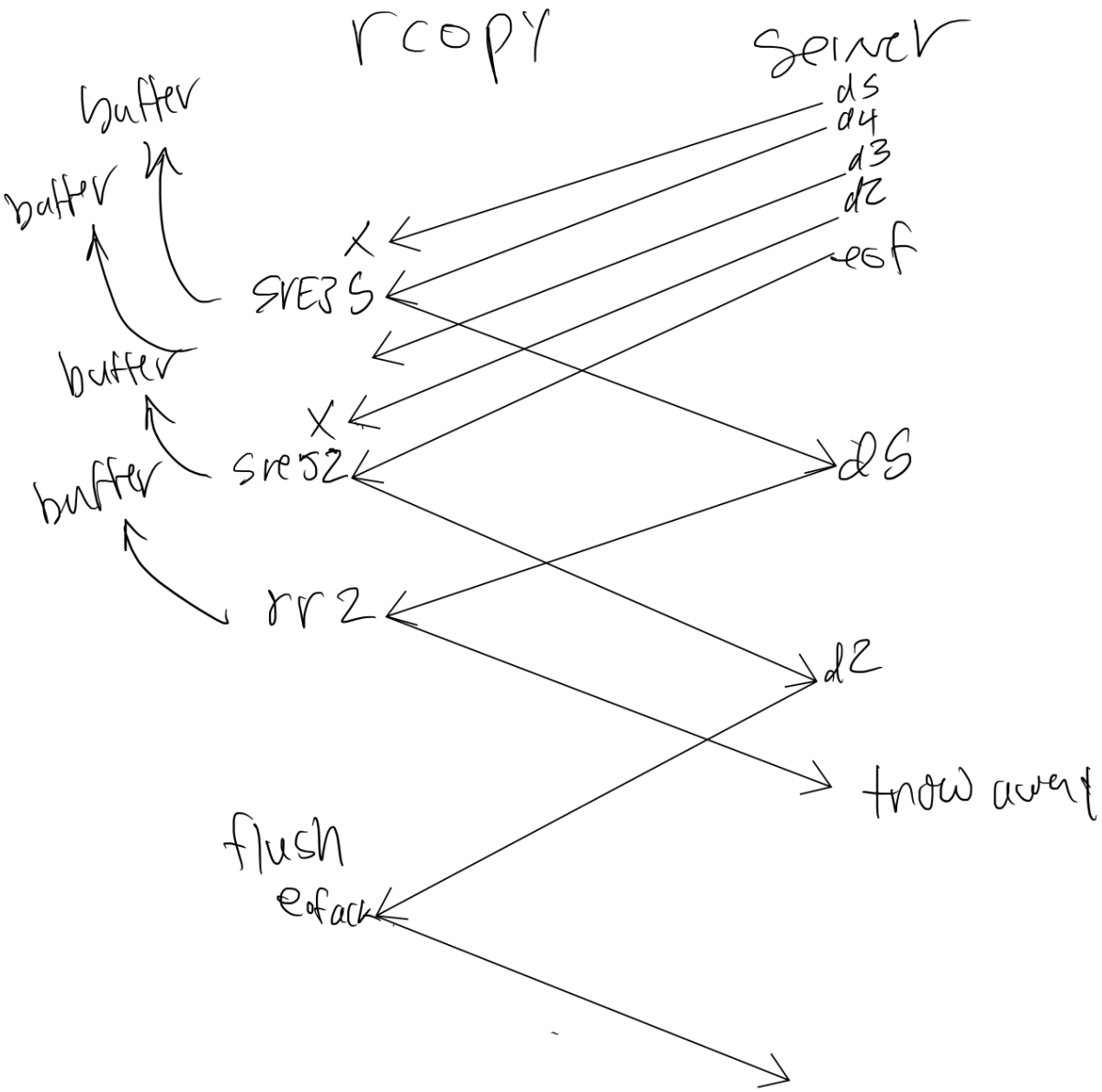
10a and 10b. Last data packet/end of transmission packet is lost



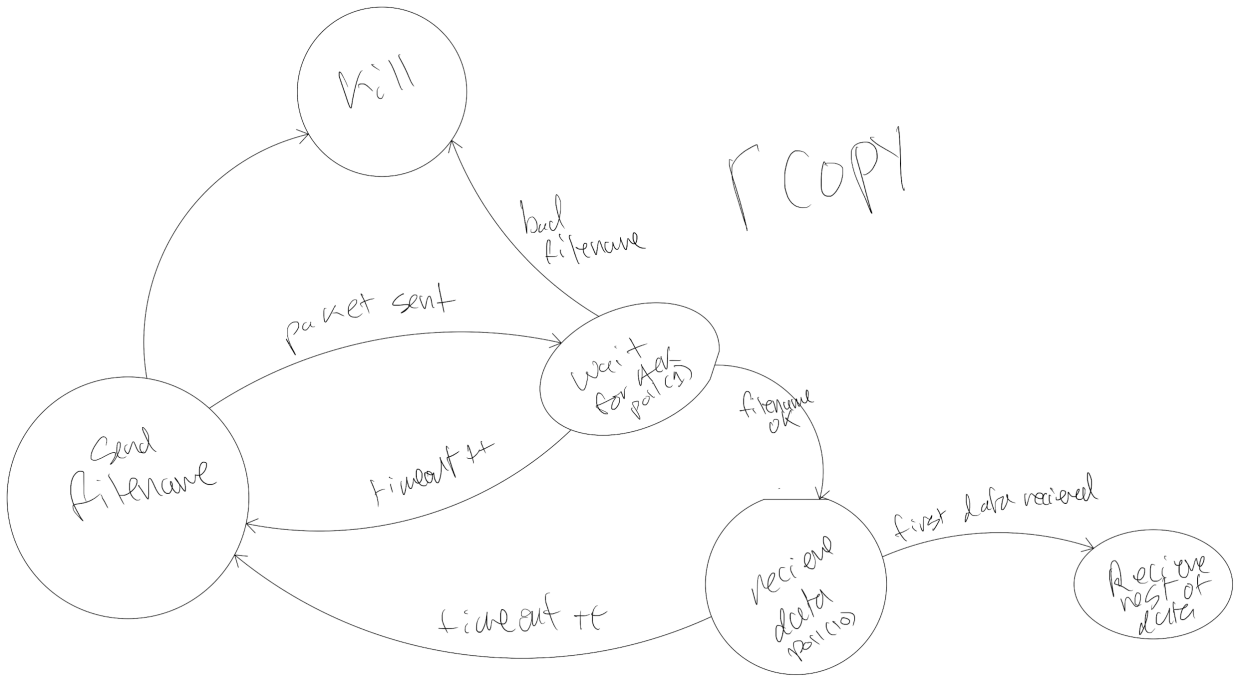
10c. RR for last packet is lost

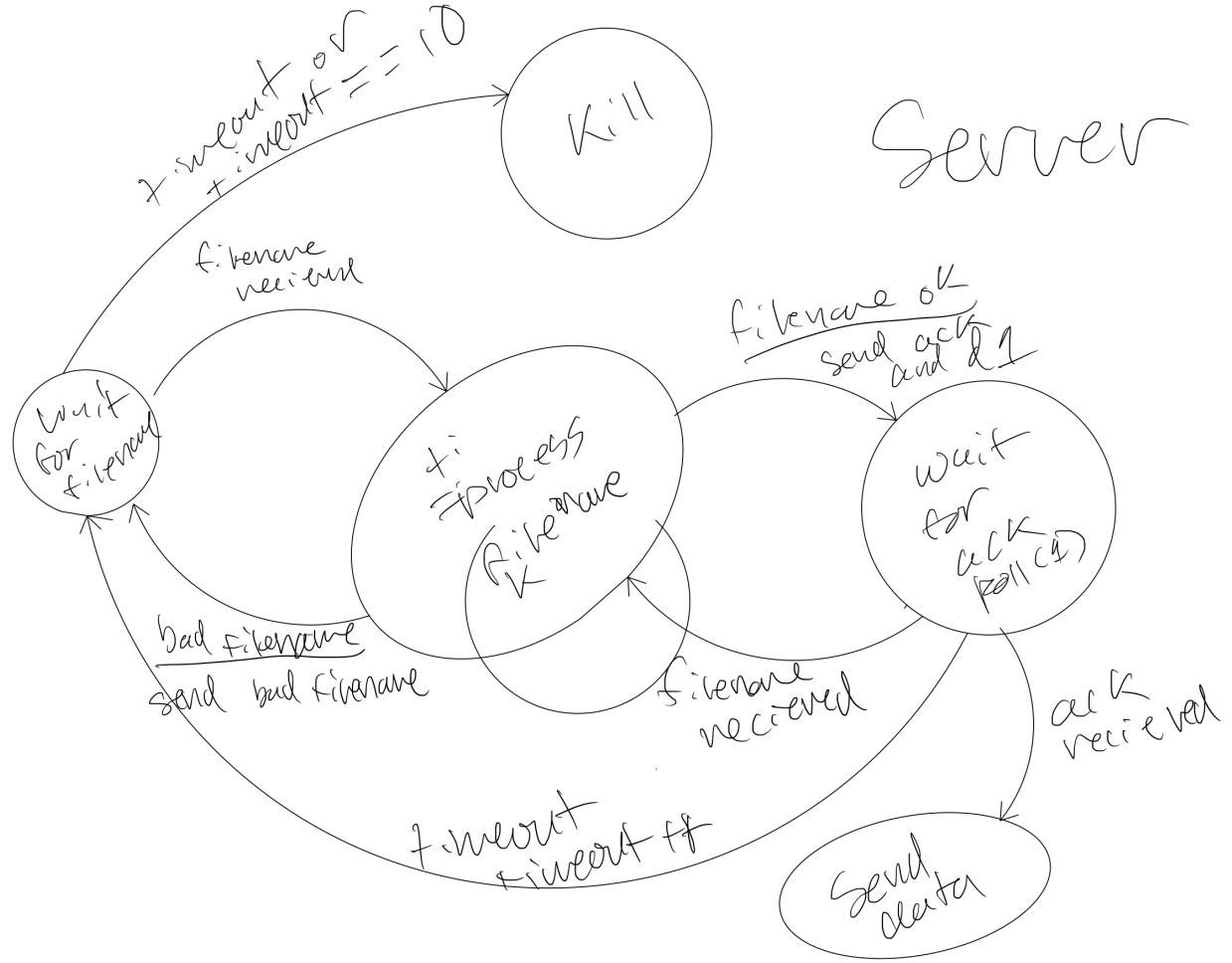


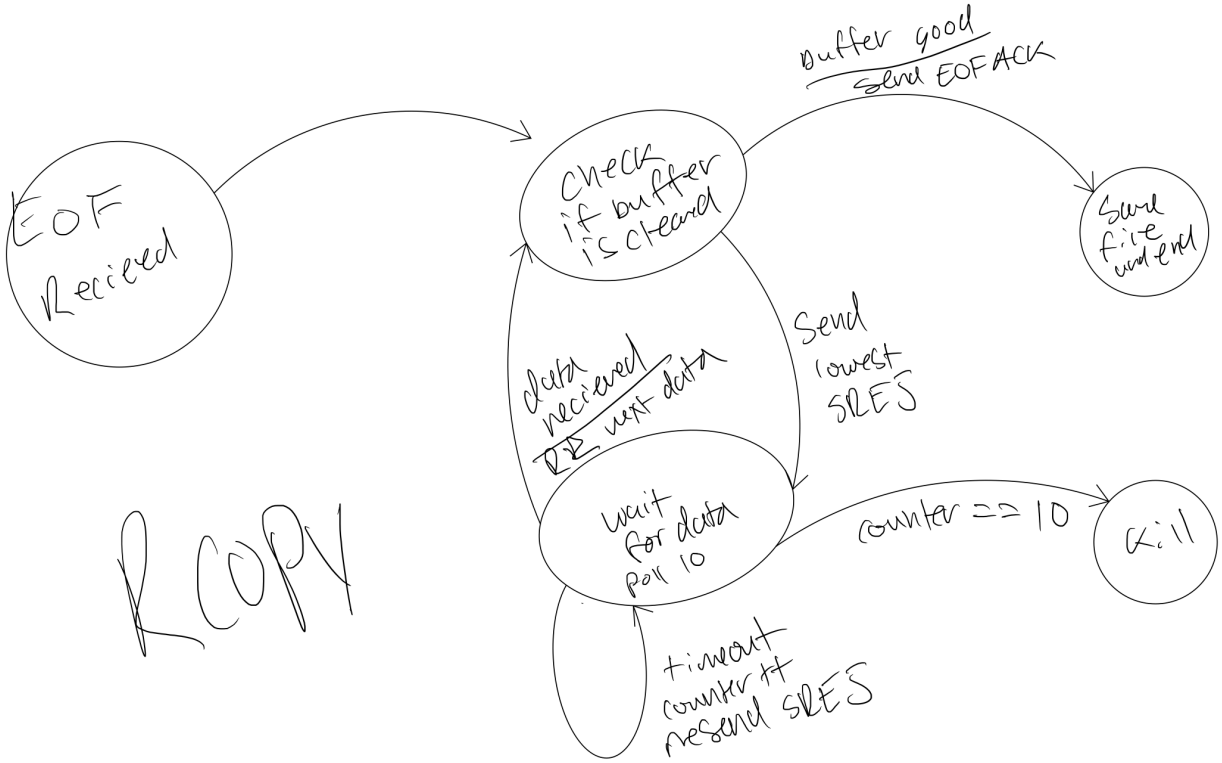
10d. 2nd and 5th packet from last are lost

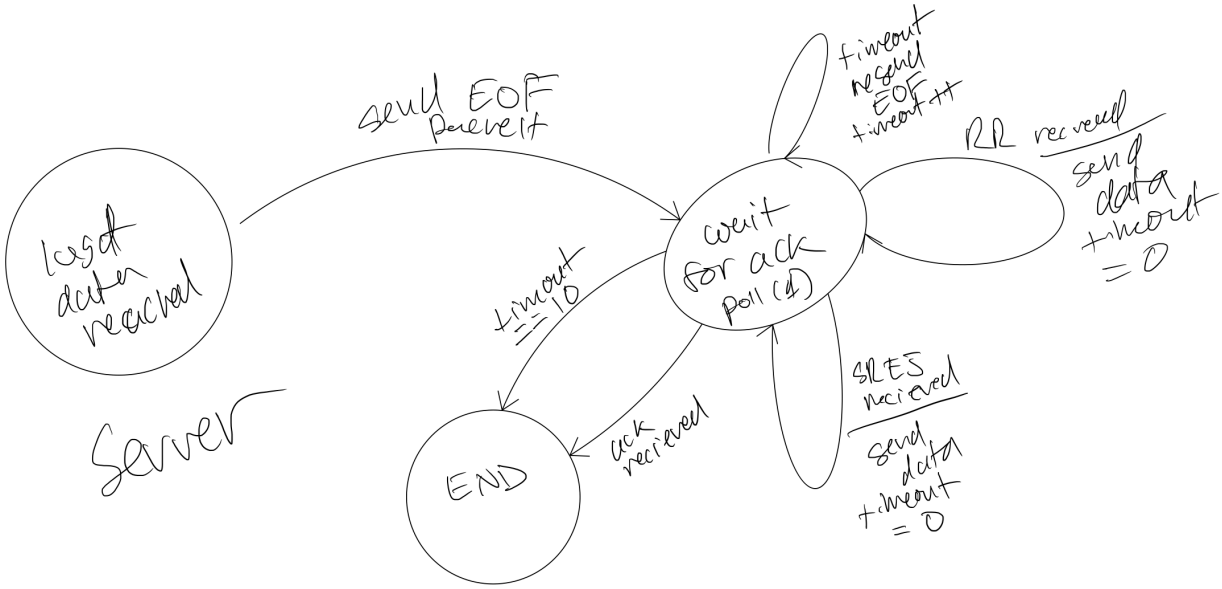


PART II - STATE DIAGRAMS


Kill
bad filename
rcopy
packet sent
wait for ack poll(1)
Send filename
filename ok
timeout ++
recieve data poll(10)
first data recieved
Recieve rest of data
timeout ++


Server
Kill
timeout or timeout == 10
filename recieved
wait for filename
filename ok
send ack and 21
ti process filename
wait for ack poll(1)
bad filename
send bad filename
filename recieved
ack recieved
timeout timeout ++
send data


EOF Recieved
Check if buffer is cleared
buffer good
Send EOFACK
Send file undone
Send lowest SREJ
data recieved
RR next data
wait for data poll 10
counter == 10
Kill
timeout
counter ++
resend SREJ
RCOPY


Send EOF
Recieve it
timeout
resend EOF
timeout ++
RR recieved
send data
timeout = 0
Last data received
wait for ack poll (1)
timout == 10
ack recieved
END
SREJ recieved
send data
timeout = 0
Server