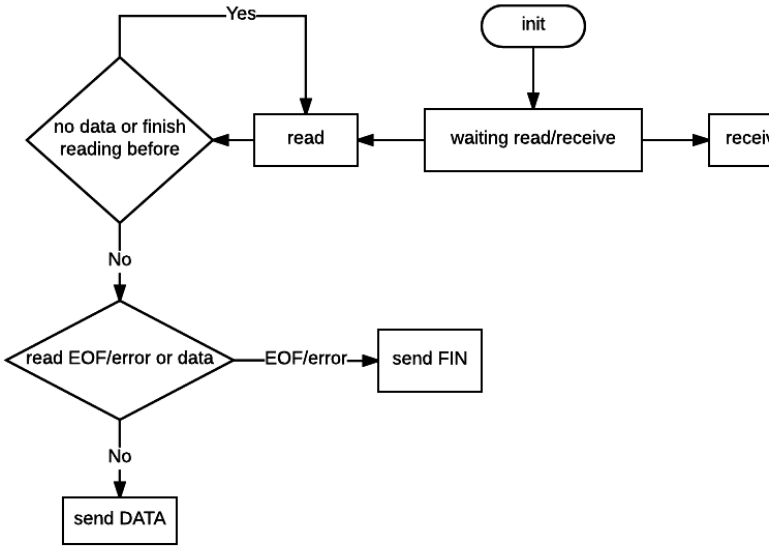
Report_1 of Lab 4

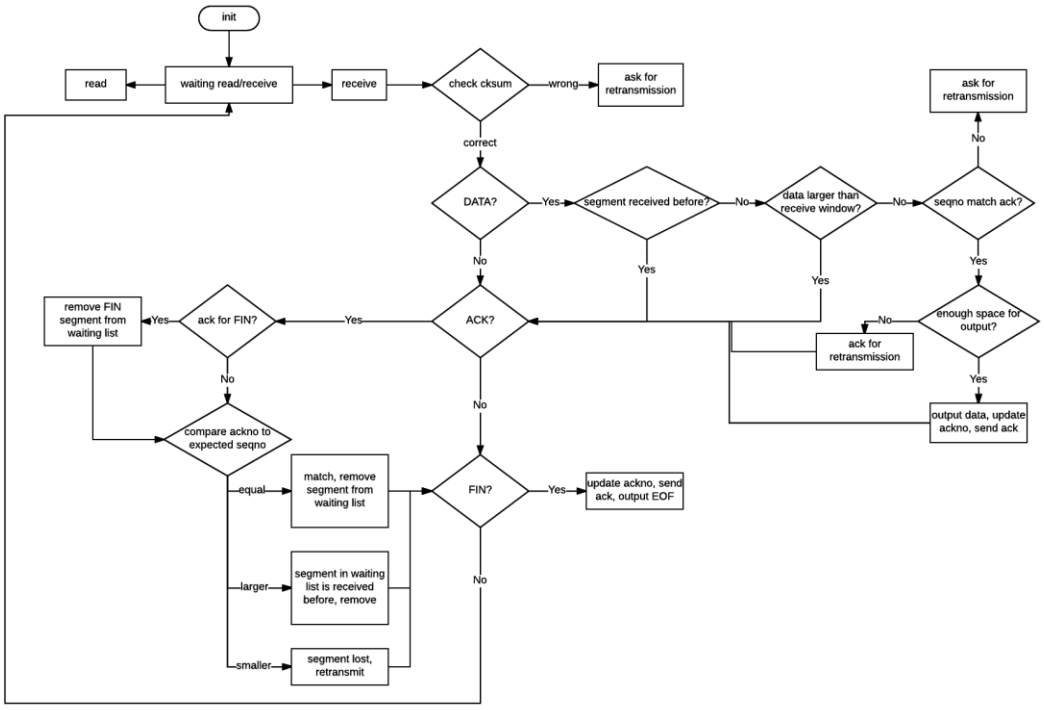
Yifan Wu

1. High-level design of cTCP implementation

Structure of read:



Structure of receive



2. Detailed implementation

For ctcp_state, add some attributes:

```
uint32_t seqno;                 /* Current sequence number */
uint32_t next_seqno;            /* Sequence number of next segment to send */
uint32_t ackno;                 /* Current ack number */
uint16_t recv_window;       /* Receive window size (in multiples of
                               MAX_SEG_DATA_SIZE) of THIS host. For Lab 1 this
                               value will be 1 * MAX_SEG_DATA_SIZE */
uint16_t send_window;       /* Send window size (a.k.a. receive window size of
                               the OTHER host). For Lab 1 this value
                               will be 1 * MAX_SEG_DATA_SIZE */
int rt_timeout;             /* Retransmission timeout, in ms */
int read_finish;            /* mark whether read is finished */
char *output;               /* bytes to output, for ctcp_output() */
int output_len;                 /* length of output */
long current_time;          /* record current time */
int FIN_received;           /* already received FIN from sender, sender disconnected */
int FIN_sent;               /* already sent FIN, disconnected to receiver*/
int get_all_ack;            /* all sent segments are acked */
int all_recv_output;        /* outpute all received segments */
int retrans_count;          /* count for retransmit times */
```

For helper functions, add

```
// send fin segment
void ctcp_send_fin(ctcp_state_t *state);
// send data segment
void ctcp_send_data(ctcp_state_t *state, char* data, int data_bytes);
// send ack segment
void ctcp_send_ack(ctcp_state_t *state);
// check cksum
int check_cksum(ctcp_segment_t *sgm);
```

In ctcp_init, init all the attributes of ctcp_state;
In ctcp_destroy, destroy the waiting list first, then free the state;
In ctcp_output, check bufspace and free the memory for the output data after call conn_output();
In ctcp_timer, retransmit segment that is time out, and tear down the connection if all conditions are achieved or retransmission of a segment exceeds 5 times.

3. Implementation challenges

The first and largest challenge is the design of cTCP state machine and flowchart. This is the most essential and important part because all coding follows the design and it takes long time to debug and redesign the whole architecture to fix the state machine and flowchart.

Then, challenges are the details in the lab description, like when to send ACK and FIN, how to tear down the connection, what's the function and return values of the given functions, how to keep a waiting list for retransmission, etc. These challenges are trivial but important, and can be overcome by reading the lab description carefully and understanding them clearly.

Besides, there are some other coding bugs like check cksum, set wrong seqno or ackno, wrongly use ntohs()/ntohl()/htons()/htonl() and so on, but these bugs can be fixed through debugging.

4. Testing cases

Since there is a tester for this lab, not too many test cases are tried.
For personally testing, I tried sending and receiving data from both client and server side, sending EOF and tearing down connection, multiple clients connecting to one server.

5. Remaining bugs

Normally using the tester to test my code, all test cases pass. However, sometimes I tried the tester then some cases failed. I'm not sure whether something wrong with my code or the tester, maybe other applications running while tester testing can affect the results of the testing.