PA3 Report

Student ID : 20211277

Name : Hojun Lee

#0. function and struct I defined or added

```
enum {

    CSTATE_LISTEN, // passive open
    CSTATE_SYN_SENT, // active open
    CSTATE_SYN_RCVD, // SYN received
    CSTATE_ESTABLISHED, // connection established
    CSTATE_FIN_WAIT_1, // active close
    CSTATE_FIN_WAIT_2, // wait for FIN from peer
    CSTATE_CLOSE_WAIT, // passive close
    CSTATE_CLOSING, // simultaneous close
    CSTATE_LAST_ACK, // wait for ACK of FIN
    CSTATE_CLOSED, // connection closed
    CSTATE_TIME_WAIT // wait for 2* maximum segment lifetime
};    /* obviously you should have more states /
```

Add more states into CSTATE enumeration.

```
typedef struct
{
    bool_t done;    /* TRUE once connection is closed */

    int connection_state;   /* state of the connection (established, etc.) */
    tcp_seq initial_sequence_num;
    tcp_seq myseqnum;
    tcp_seq peerseqnum;
    tcp_seq last_peer_ack; //

    bool_t ack_pending;
    int packets_since_ack;
    /* any other connection-wide global variables go here */
} context_t;
```

In context_t struct, I add some variables for representing peer’s seqnum, acked number, and whether now ack is pending.

```c
static void send_ack(mysocket_t sd, context_t *ctx){
    char send_buf[sizeof(struct tcphdr)];
    struct tcphdr* send_hdr = (struct tcphdr*) send_buf;

    init_tcphdr( send_hdr, ctx->myseqnum, ctx->peerseqnum, TH_ACK);
    stcp_network_send( sd, send_hdr, sizeof(struct tcphdr), NULL );
}

static void init_tcphdr(
    struct tcphdr* hdr,
    tcp_seq seq_num,
    tcp_seq ack_num,
    uint8_t th_flags
){
    memset(hdr, 0, sizeof(struct tcphdr));
    hdr->th_seq = htonl(seq_num);
    hdr->th_ack = htonl(ack_num);
    hdr->th_off = 5; // Data Offset
    hdr->th_flags = th_flags;
    hdr->th_win = htons(WINDOWSIZE);
}
```
Send_ack function is a function sends ack. init_tcphdr construct tcp header.

#1. 3-way handshaking

In transport_init function, is_active value represents whether it is client or server. Depending on it, 3-way handshaking is done by using stcp_network_recv, stcp_network_send function.

#2. Data transfer & 4-way handshaking

After 3-way handshaking, data transfer and 4-way handshaking is done in control_loop.

Stcp_wait_for_event function check queues for app data, network data and timeout occur. If one have sent WINDOWSIZE data and have not received the ack then it must skip APP_DATA event. Thus, NETWORK_DATA event should be treated first. .And then if the data from APP_DATA exists it can be sent.
when NETWORK_DATA event occurs, first check the header and flag.

When APP_CLOSE_REQUESTED event occurs meaning active close.

Peer who receives that packet from network layer is passive close.

If a host which is FIN_WAIT_1 state receives FIN packet then it means both are closing, (i.e. simultaneous close) then they changes states to closing and close when receive ack packet.