

Lab08 Robust UDP Challenge

LiuLiu

109550146 劉沛凡

109550143 劉仲恩

Result

- latest result, now in rank 27

```
** checking files ...
[0] ..... [10] ..... [20] ..... [30] ..... [40] ..... [50] .....
    [60] ..... [70] ..... [80] ..... [90] ..... [100] ..... [110]
0] ..... \x1b[1;32m. [120] ..... [130] ..... [140] ..... [150] .....
. [160] ..... [170] ..... [180] ..... [190] ..... [200] ..... [210]
0] ..... [220] ..... [230] ..... [240] ..... [250] ..... [260] ..
..... [270] ..... [280] ..... [290] ..... [300] ..... [310] .....
... [320] ..... [330] ..... [340] ..... [350] ..... [360] ..... [
370] ..... [380] ..... [390] ..... [400] ..... [410] ..... [420]
..... [430] ..... [440] ..... [450] ..... [460] ..... [470] .....
..... [480] ..... [490] ..... [500] ..... [510] ..... [520] .....
[530] ..... [540] ..... [550] ..... [560] ..... [570] ..... [580]
] ..... [590] ..... [600] ..... [610] ..... [620] ..... [630] ...
..... [640] ..... [650] ..... [660] ..... [670] ..... [680] .....
.. [690] ..... [700] ..... [710] ..... [720] ..... [730] ..... \x1
b[m [740] ..... [750] ..... [760] ..... [770] ..... [780] ..... [
790] ..... [800] ..... [810] ..... [820] ..... [830] ..... [840]
..... [850] ..... [860] ..... [870] ..... [880] ..... [890] .....
..... [900] ..... [910] ..... [920] ..... [930] ..... [940] .....
[950] ..... [960] ..... [970] ..... [980] ..... [990] .....
** size incorrect = 0 , corrupted files = 0
** submitting to scoreboard ...
ok
** client runs for 172.348924 second(s)
** success rate = 1.000000 ; files = 1000 / 1000
```

Parameters & Data Structures

```
#define MAXLINE 512
#define SEND_TIME 24
#define CHECK_ACK_TIME 256
#define MAX(a, b) (a>b?a:b)

typedef struct {
    int    file_no;
    int    seq_no;
    int    eof;
    char   data[MAXLINE];
} pkt_t;

typedef struct {
    int file_no;
    int seq_no;
} ack_t;
```

The idea about send and receive

- For client
 - Suppose no packets are lost during transmission
 - Only resend after receiving ack
- For server
 - Only receive in-order packets
 - Send ack every certain period of time

Client

- for all file, read the content into packets (size : 1024 bytes)
- for each packet, send it to server for at most 32 times. After sending to server, client wait for a while for ACK response.
- If received ACK, move on to next packet.
- After all packets are sent, send an EOF to server

Client (cont.)

- main loop look like this

```
while (read(fr, pkt.data, sizeof(pkt.data)) > 0) {
    for (int i = 0; i < SEND_TIME; i++) {
        if(sendto(s, (void*) &pkt, sizeof(pkt), 0, (struct sockaddr*) &sin, sizeof(sin)) < 0)
            perror("sendto");
        // recv ack
        usleep(50);
        int rlen;
        if((rlen = recvfrom(s, (void*) &ack, sizeof(ack), MSG_DONTWAIT, NULL, NULL)) > 0){
            //printf("RECV: ");
            //printack(&ack);
            if(ack.file_no == pkt.file_no && ack.seq_no == pkt.seq_no+1){
                break;
            }
        }
        usleep(100);
    }

    pkt.seq_no++;
    seq_no++;

    memset(&pkt.data, 0, sizeof(pkt.data));
}
```

Server

- After receiving packets and EOF, send ACK indicating the next packet that client should send
- only write to file if receive the valid packet

Server (cont.)

- main idea of server

```
/* Deal with packets */
pkt_t pkt;
for ( ; ; ) {

    /* Receive packet */
    if((rlen = recvfrom(s, (void*) &pkt, sizeof(pkt), 0, (struct sockaddr*) &csin, &csinlen)) <= 0) {
        //usleep(1);
        /* store */
        // write to different file?
        if (file_no == pkt.file_no && seq_no == pkt.seq_no)
        {
            if (pkt.eof == 1) {
                file_no++;
                seq_no = 0;
                break;
            }

            if (write(fw, pkt.data, strlen(pkt.data)) < 0)
                perror("write");

            seq_no++;
        }
        ack_t ack = {.file_no = file_no, .seq_no = seq_no};
        sendto(s, (void*) &ack, sizeof(ack), 0, (struct sockaddr*) &csin, sizeof(csin));
    }
}
```