Sangjoon Lee 004 749 872
William Lee 904 814 655

CS 118 Project 2 Description

Implementation description

header format: (4 byte)

1. Sequence number (16 bit) : sequence ID ranges from 0 to 30720
2. Flag (4 bit) : each digit representing ack, syn, fin, 404 in that order
3. Size (12 bit) : size of the entire packet (max 1204)

Messages:

- Aside from client sending requested file name in the very beginning, messages (or payload) only contain data of the requested file.
- Header (4B) + payload (1020B) makes up max package 1024B
- After finished reading, the file, stop sending packets

Timeouts:

- Kept track of real time in milliseconds for each packet in the window, and upon timer expiration for a certain packet, resend that packet.

Window-based protocol:

- Server side:
    - At the start, server waits to receive filename from the client.
    - Upon getting filename, server checks if file exists and sends 404 or will continue with a SYN.
    - Each sequence number is in bytes and incremented by the number of bytes sent in each packet (including header).
        - On reaching the max sequence number, reset sequence to 0
    - Window for storing 5 packets where each ACKed packet is marked and window is advanced when ready.
- Client side
    - Client waits to receive packet
    - Upon getting packet, stores the sequence number and payload (stores only up to 5 packet in window)
    - Sends ack with same number as sequence number.
    - Only when packet with base seq number arrive it, it starts writing in the file upto where it can. Those it wrote, it removes from the window.

Difficulties

- UDP protocol
    - Sockaddr_in failed to work when sending packets in server side
        - Apparently Sockaddr_in used to receive from UDP socket was not enough for transmitting file and we had to use “sockaddr_storage”
- Server side (SR_server.c)
    - When alternating between sending messages and receiving them (ACK, SYN, FIN), had to make sure that it matched with client so that both don’t get stuck in waiting when Introducing packet loss
    - Finding out that I was using the wrong data type when shifting arrays for the timer which made retransmission not happen
    - Making sure that when sending data, I need to wait for the client side because there would be packet loss
    - For handling the last packet for the last block of the file (which may be less than a full payload packet), I just read until file is completely read and transmitted once file is completely read.

- Client side (SR_client.c)
    - Keeping track of window; values inside window was kept changing and corrosponding ack meant different payload. Normal array and index wouldn’t have worked since that meant having index from 0 to 30720.
        - Decided to create entire structure to hold all info regarding a sequence number in window (including payload). This kept all payloads in one place and allowed me to keep track of sequence using fields instead of indexes.
    - Duplicate seq ID; I needed to keep track of given sequence IDs for printing “retransmission” for retransmitted ACK. I had to make sure wrap around sequence number didn’t count as candidate for retransmission ACK
        - Solved by dividing the sequence range into 3 part and keep erasing part of the range ahead of the one I am currently writing to solve wrap around duplicates.