2140 Hjemmeeksamen DESIGN-dokument

I denne oppgaven så har jeg lagd en fil for klient som heter «upush_client.c» og en fil for server som heter «upush_server.c». Begge disse filene har en header og en c-fil med funksjoner som brukes i disse filene (client_funcs.c og server_funcs.c). Jeg har også lagd en header og c-fil for funksjoner og structer som er felles for begge, denne heter common. Dette er gjort for at det skal bli mer oversiktlig.

I klient og Server har jeg valgt å oppbevare klienter i et size_t array hvor jeg lagrer pekerene til client-structene castet til size_t. For å hente klienter ut så caster jeg de tilbake til struct client peker. Jeg starter ved å la arrayet ha plass til 10 klienter og hvis antall klienter kommer til å overskride dette så bruker jeg realloc () til å lage plass til dobbelt så mange klienter. Jeg har også valgt at hver pakketype har egne sekvens nummere som teller oppover fordi det da er lettere å se hvor mange pakker som er sendt.

Jeg har med flere fprint-setninger i programmet slik at det skal bli mer oversiktlig å rette og se at det funker.

Client:

Hendelsesløkken i «upush_client.c» starter med å sjekke om det skal bli foretatt en registrering (har valgt å bruke registrering som heartbeat-melding også) hvis det har gått 10 sekunder siden siste melding. Første iterasjon blir det alltid foretatt en registrering. Hvis det har gått mer enn 30+timeout sekunder så har jeg valgt å avslutte programmet og printer «ingen kontakt med server» så den ikke skal fortsette å sende meldinger. Det er + timeout for de tilfellene der timeout er mer enn 30 sekunder. Det blir sendt en heartbeat/REG melding til server og det blir sjekket om denne meldingen har fått tilbake ACK innen timeout. Det vil si at ved pakketap der timeout er stor så blir ikke en REG-melding sendt hvert 10.sek, men jeg har antatt at dette er sånn oppgaven skal gjøres.

Jeg har i denne oppgaven valgt at select skal vente maksimalt timeout 10 ms, dette er valgt for å få en liten ventetid ved select. Dette gjør at presisjonen på når meldinger blir sendt på nytt i henhold til timeout er bedre fordi den da oftere sjekker om det er noen meldinger som ikke har mottatt ACK i løpet av timeout perioden.

Når det kommer noe inn fra STDIN så blir denne meldingen analysert og hvis den har riktig format blir melding sendt til klienten og lagret i en lenkeliste av meldinger sendt til denne klienten i klient-structen. Meldingene blir lagret i en slik rekkefølge at den meldingen som er lengst siden ble lagt inn blir alltid prøvd å bli sendt først (FIFO) fordi det er slik brukere vil at meldingene skal ankomme mottakere. Selv om dette ikke blir tilfellet dersom mange pakker går tapt av en bestemt melding (da vil denne bruke lengst tid). Hver gang det skjer en timeout så blir den eldste pakken til hver klient sjekket om det er mer enn 2 sekunder siden sendt og hvis det er det så blir den sendt på nytt. Tiden den ble sendt på og en tries-teller blir oppdatert for å vite hva som skal gjøres med meldingen. Melding mellom klienter blir printet til teminalen på følgende form: **[Fra_nick] tekst.**

Server:

Jeg har valgt å implementer select i «upush_server.c» som lytter til sock og STDIN_FILENO for å ha en mulighet til å avslutte server og free'e alt minnet som ble brukt av heapen. For å avslutte server så taster man inn samme kommando som i klient: «QUIT».

Når det har gått over 30 sekunder siden forrige heartbeat, så blir klienten slettet fra REG_clients ved at porten til klienten blir satt til 0. Når en annen klient da neste gang skal bli satt inn så kan den bli satt inn på plasser hvor klientens port er 0.

Generelt: Jeg har valgt å bruke andre måter å dele opp strenger på enn å bruke strtok() for å vise litt mer forståelse. Det er en av grunnene til at det har blitt ganske mange kodelinjer.

Har valgt å skrive de fleste informasjons utskrifter og feilmeldinger til stderr og utskrifter som bruker trenger for å kjøre bruke programmet (f.eks [fra_nick] tekst osv) til stdout. Dette er for at retter enklere kan skille filtrere ut utskrift.

Alle implementasjoner som oppgaveteksten beskriver, skal fungere.