kandidatnummer: 15617

Designdokument for UPush chattjeneste

For å få til dynamisk allokering av klienter på serveren, klienters bekjente klienter (heretter "kontakter"), meldingskøer og blokkerte nicker valgte jeg å bruke lenkelister. Jeg valgte å utfordre meg selv litt ved å lage en generisk lenkeliste, som så kunne brukes for å lage spesialiserte implementasjoner. Det tok en stund, men den funker bra.

For å kommunisere mellom tjener og klienter valgte jeg å implementere en stopp-og-vent-protokoll med 0 og 1, sånn at sender av en ny melding (ikke en ACK), venter til mottatt ACK før den sender neste melding til samme mottaker. Dette funker også bra, men førte til en problem jeg ikke fikk løst ved brukere som kobler fra og til igjen.

(Jeg tolket oppgaven som at klienter i utgangspunktet kun må slå opp andre klienter på server i det at de prøver å sende meldinger selve, eller de skal sende en ACK til en mottaker de ikke kjenner.)

Hvis en klient A sender PKT 0 til en klient B, vil den vises på klient B sin skjerm. B sender ACK tilbake, A justerer slik at neste melding vil få 1, og nå forventer B at neste melding fra bruker med samme nick som A vil være 1. Hvis A da reconnecter, vil den nullstille seg, og sende 0 igjen. B vet ikke at A har reconnectet, og tenker at ACKen den sendte nok gått tapt, og viser ikke den nye meldingen for å ikke forstyrre bruker. Det er altså en svakhet i at B kun sjekker nicken til avsender ved nye meldinger. Hvis den også sjekket adresse, eller slått opp nicken på server på nytt ville dette nok funket bedre.

I både client og server brukes noen globale variable for å holde på de viktigste datastrukturene. For klienter er dette sin nick, sin input timout, kontakt-listen, serveren, blockerte nick-listen og sin socket fd. For server er dette sin klient-liste og sin socket. Dette gjør det smidig å bruke dem der man trenger dem. En ulempe med måten jeg laget systemet på er at det nok ikke er delt opp i veldig tydelige lag. Hvis jeg hatt mer tid ville jeg jobbet mer med det grensesnittet.

Jeg har prøvd å sjekke alle returverdier jeg klarte, men programmet terminerer ved minste feil, som kanskje ikke er det beste. Det kunne kanskje vært hensiktsmessig å ikke quitte helt bare fordi gettimeofday mislykkes kun én gang, så fremt det ikke er kritisk.

Jeg redefinerer SIGINT i begge programmer for å sørge for at allokert minne blir frigjort før programmene terminerer.

Jeg valgte å la select-kallet i hovedhendelsesløkken få timeout etter hvert halve sekund, sånn at vi skal få sendt hjerteslag og sjekket opp om det er meldinger vi må resende ofte.

I det store hele har systemet funket godt når jeg testet med 3 klienter og skriver så raskt jeg kan, men jeg merket litt merkelig beteende hvis det er stor forskjell i timeout-input på de forskjellige klientene. Da ender fort en klient opp med å ha fått meldinger at avsender ikke skjønner er levert.