UDVIKLING AF PROTOCOL STACK - FILOVERFØRSEL VIA RS-232 NULL-MODEM

I denne øvelse skal designes og implementeres mulighed for at overføre en fil vha. den serielle kommunikations-port i en virtuel maskine. Det serielle interface er i denne øvelse et RS-232 interface. Microcontroller-baseret embedded udstyr har ofte kun mulighed for at kommunikere med omverdenen via et serielt interface. Derfor er problematikken i denne øvelse relevant. Det er desuden yderst lærerigt at udvikle en protocol stack, hvilket er hovedformålet med øvelsen.

Systemet skal give mulighed for at der fra en virtuel computer (H1) kan overføres en fil af en vilkårlig type/størrelse til en anden virtuel computer (H2).

Der skal designes, implementeres og testes to applikationer, en client og en server. Førstnævnte kan anbringes i H1, sidstnævnte kan anbringes i H2.

Den ene applikation (client) skal meddele serveren hvilken fil (evt. incl. sti-angivelse) der skal hentes.

Den anden applikation (server) skal læse og sende filen til klienten i pakkestørrelser på 1000 bytes payload ad gangen.

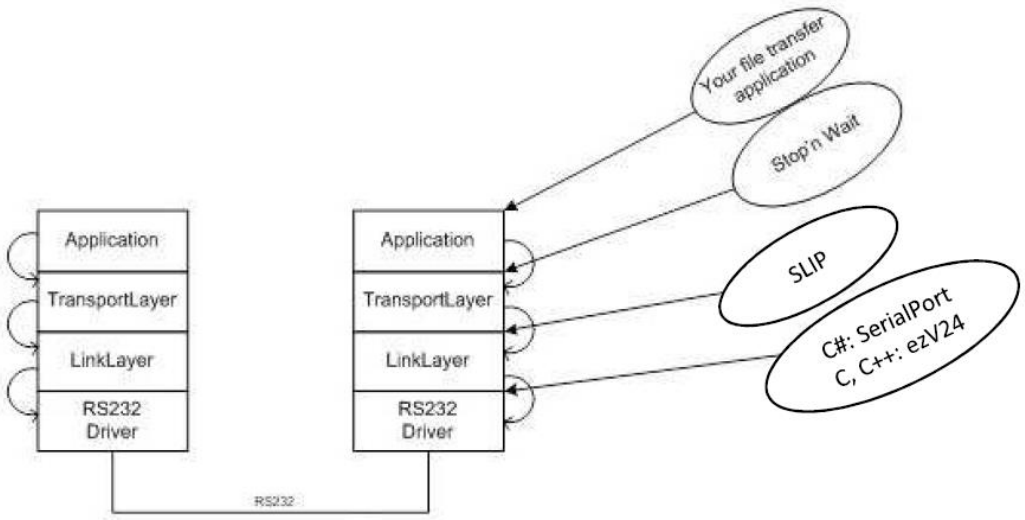
Client skal modtage disse pakker og gemme dem i en fil.

Såvel server som client er nemme at realisere, idet disse ”applikationslags-applikationer” allerede er udviklet i øvelse 7 (TCP-baseret client/server).

I øvelse 7 blev der vha. anvendelse af socket-API udført kald som etablerede en connection, overførte filnavn, filstørrelse og fil mellem client og server. Disse kald skal i denne øvelse udskiftes med kald til et transportlaget i en protocol-stack, som I selv udvikler. Transportlaget skal være pålideligt med en funktionalitet, som svarer til rdt v.2.2 i lærebog/slides.

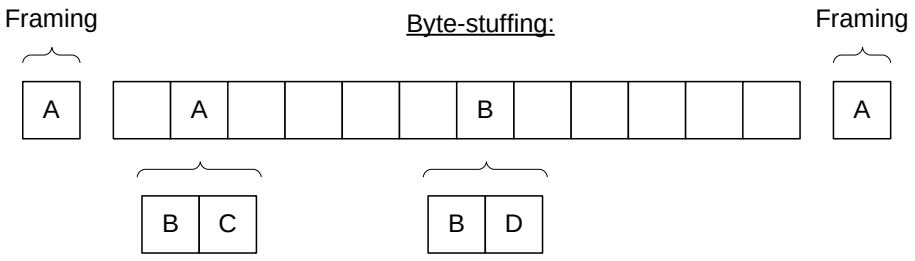
Client og server kan også have samme brugerflade som client-/server-applikationerne i øvelse 7 (TCP/IP-baseret filoverførsel).

Opgaven løses ved at dele de enkelte funktioner efter en lagdelt model for en protokolstak:



1) **<u>Det fysiske lag:</u>** Dette lag er givet ved seriel kommunikation via RS-232 porte og via et null-modem (/dev/ttyS1).

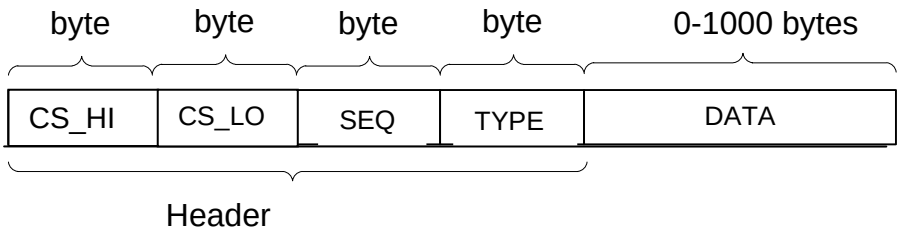
2) **<u>Datalinklag:</u>** Til dette lag skal I implementere en modificeret SLIP protokol. Protokollen skal implementeres således: Som *start* og *stop* karakter benyttes 'A'. Hvis karakteren 'A' forekommer i *frame't* erstattes det med de to tegn 'B' og 'C', og hvis tegnet 'B' forekommer, erstattes det med de to tegn 'B' og 'D'.



3) ***<u>Netværkslag:</u>*** Dette lag findes ikke, da vi kun anvender point-to-point kommunikation (dvs. kommunikation uden routing).

4) ***Transportlag:*** I skal implementere en stop-and-wait protokol. Protokollen skal være pålidelig og skal anvende en 16 bit Internet-checksum til fejldetektering. Det er et krav at protokollen skal kunne håndtere ødelagte data (f.eks. pga. EMC). Det er *ikke* et absolut krav at protokollen skal kunne håndtere mistede data (timeout, rdt v. 3.0 ifølge lærebogen) men implementer timeout-funktionaliteten hvis tiden er til det - få stop-and-wait protokollen til at fungere først. Den maksimale længde af transportlagets data (payload) sættes til 1000 bytes.

   Dette lag skal overholde følgende segment header format:



CS_HI: Den mest betydende del af checksums-beregningen.
CS_LO: Den mindst betydende del af checksums-beregningen.
SEQ: Sekvensnummer på den afsendte segment.
TYPE: 0 (DATA)
1 (ACK)
DATA: Payload på mellem 0-1000 bytes. Skal altid være af længden 0 ved ACK.

5) **Applikationslaget**: Her skal client sende et filnavn (defineret af brugeren) til server – og server skal returnere filstørrelse og selve filen til client via transportlaget. Der kan anvendes meget kode fra Øvelse 7.

Hints:

1) I udviklingsforløbet kan I anvende en simuleret seriel port mellem to virtuelle maskiner, som kører i den samme laptop. Vejledning i opsætning af den simulerede port findes på Blackboard, I4IKN, Link Layer i filen: *Connecting_Two_Virtual_Machines_via_serial_ports.pdf*

2) Det kan anbefales først at implementere SLIP protokollen i både afsender- og modtager-applikationen og derefter teste denne protokol vha. testdata, som vises på modtagerens skærm. Test med ”synlige” bytes vha. terminalen *minicom*, som kan downloades og installeres i de virtuelle maskiner

3) Den nemmeste måde at teste det færdige programmel på, er at prøve at overføre en ”ikke tekst-”fil, f.eks. et stort .jpg-billede. Brug Linux-terminalkommandoen *cmp* eller *diff –s* til at verificere at de to filer er ens.

Dokumentation:

> Beskriv udviklingsforløbet, funktionaliteten og resultatet i en journal i størrelsesordenen 6 A4 sider incl. relevante små udsnit af jeres source-code . Afleverer desuden jeres komplette source code i en zip-fil. Afleveringen foregår på Blackboard, I4IKN på samme måde som øvelse 7/8.
>
> Journalen for øvelse 12 skal demonstrere forståelsen for udviklingen af en protocol-stack, der kan håndtere pålidelig kommunikation. Godkendelse af journalen er (som for øvelse 7/8's vedkommende) en betingelse for at blive indstillet til I4IKN-eksamen.

Gennemførelse:

Programudvikling og udfærdigelse af journal skal foregå i grupper på 2-4 studerende.

Demonstration:

Grupperne demonstrerer over for underviseren at client/server-opstillingen og protokol stack'en fungerer efter hensigten.