Løsningsforslag

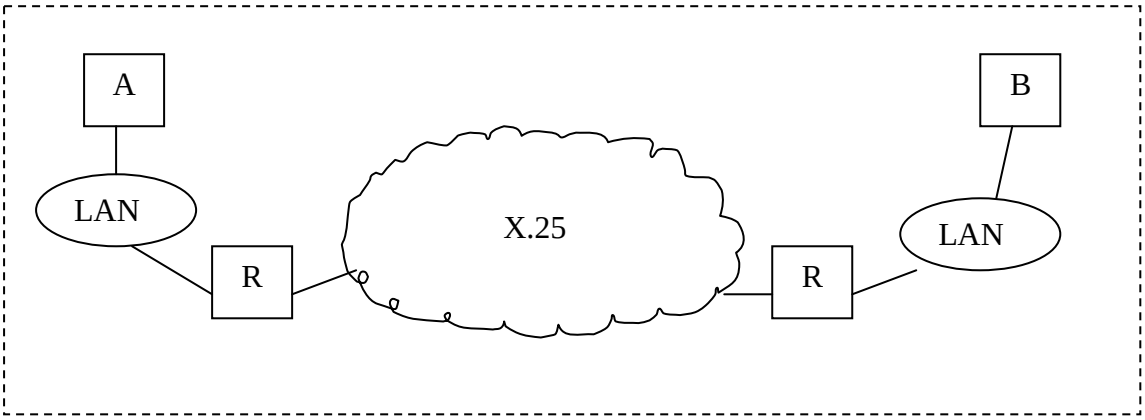
UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i	IN 270 - Datakommunikasjon
Eksamensdag:	Torsdag 7. juni 2001
Tid for eksamen	9.00 - 15.00
Oppgavesettet er på	4 sider
Vedlegg:	Ingen
Tillatte hjelpemidler	Alle trykte og skrevne hjelpemidler, og kalkulator

Kontroller at oppgavesettet er komplett før du begynner å besvare spørsmålene
Husk å skriv tydelig og lesbart. Gi kortest mulige svar, ikke lange utlegninger.
Dersom du på noe punkt finner oppgaveteksten uklar kan du gjøre dine egne presiseringer. Gjør i så fall tydelig rede for disse i besvarelsen din.

1. Internett kommunikasjon via et forbindelses-orientert nettverk



Internett kommunikasjon via et forbindelsesorientert nettverk

Vi skal i denne oppgaven se på sammenkoplingen av to lokalnett av Ethernet typen via et forbindelsesorientert nettverk, for eks. X.25, og ønsker svar på følgende spørsmål:

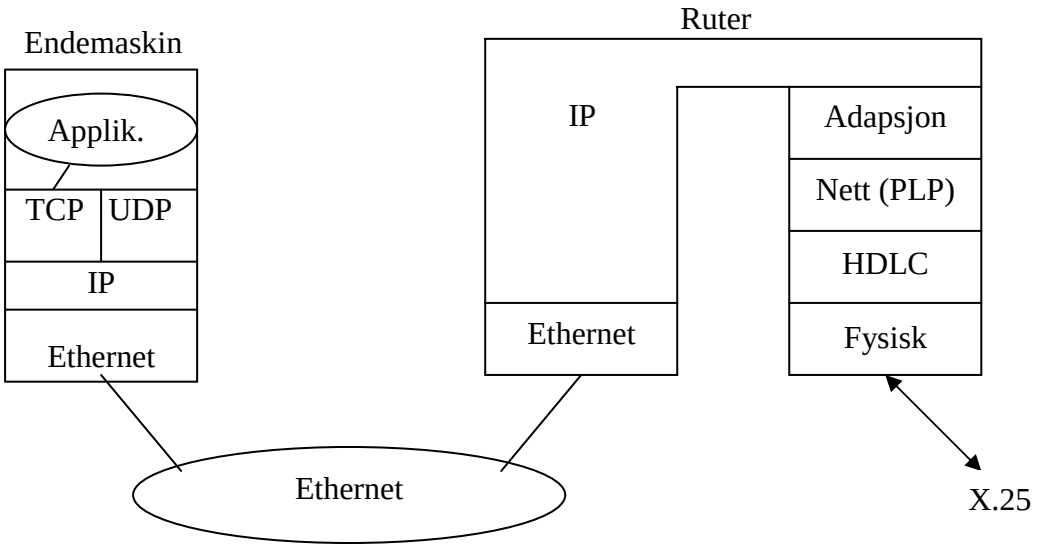
a. *Beskriv i grove trekk tjenestene det forbindelses-orienterte nettet tilbyr, i form av et sett prosedyrekall eller kommunikasjonsprimitiver. Angi noen av de viktigste parametrene i hvert kall.*

Prinsippet for forbindelses-orientert kommunikasjon er behandlet i kurset, men ikke X.25. Men studentene skulle klare å resonere seg frem til svarene på spørsmålene.

Tjenesten nettlaget tilbyr kan i grove trekk beskrives ved hjelp av tjenesteprimitivene:

N-Connect; med som et minimum følgende parametere: N-addr, N-SAP; retur= Ref,
N-Disconnect; med parameter Ref,
N-Send; med parametere: Ref, peker til pakkebuffer, buf-len,
N-Rcv; med parametere: Ref, peker til tomt pakkebuffer; retur=buf-len.

b. *Tegn protokollhierarkiet i ende-maskinene, for eks. i A og i en av ruterne, og forklar funksjonaliteten til hvert lag i hierarkiet.*



De viktigste funksjonene til lagene illustrert i figuren over:
Ethernet: pakker inn en IP-pakke i en Ethernet-pakke med en MAC-adressen til mottaker spesifisert av IP-laget. Ved kollisjon, prøver Ethernet laget å sende pakken om igjen etter en viss vilkårlig ventetid. Ved gjentatte kollisjoner, dobles ventetiden (eksponsiell backoff). Opp til 16 forsøk før Ethernet laget gir opp. Ethernet laget aksepterer pakker på grunnlag at MAC adressen i pakken. Sjekksummen verifiseres, og hvis OK, leveres nyttelasten (IP pakken) opp til IP laget. Ethernet tjenesten er en ren datagram tjeneste, så her er ingen kvitteringer.
IP-laget: tilbyr en ren datagram tjeneste til laget over og bygger på en ren datagram tjeneste fra laget under. IP-laget benytter globale adresser, og fremsender pakker på grunnlag av mottakers IP-adresse. I tillegg til ruting og fremsending, så er den viktigste funksjonen i IP-laget fragmentering og reassemblering. IP-laget er implementert i alle endemaskiner og rutere.
TCP: tilbyr en pålitelig forbindelses-orientert tjeneste til applikasjonen, men bygger på en ren datagram tjeneste fra laget under (IP-laget). Benytter glidende vindu teknikk med dynamisk justerbart vindu for flytkontroll. I tillegg kombineres dette med en dynamisk metningskontroll, for å redusere metning i nettet. TCP interpreterer tap av kvitteringer som metning, eller begynnende metning, i nettet, og reduserer sitt dynamisk justerbare vindu.
UDP: tilbyr omtrent sammen tjenestekvalitet som IP-laget, men gir muligheten for multipleksing.
Fysisk lag: er spesifikk for ulike typer nett. Oppgaven er å modulere digital informasjon inn på transmisjonsmediet, slik at dette lar seg transporterer gjennom transmisjonssystemet og at mottakersiden er i stand til å gjenskape den digitale informasjonen mest mulig korrekt. Dette impliserer at mottakersiden må settes i stand til, ad ulike metoder, å synkronisere seg til sendersidens bit-takt. Tjenestetilbudet er et bit-orientert grensesnitt for sending og mottaking av bit.
HDLC: HDLC er et eksempel på et linklag. Det samler bit i rammer for å oppdage og korrigere for transmisjonsfeil via sjekksumtest og utøver flyt kontroll. HDLC tilbyr en forbindelses-orientert tjeneste til laget over, og benytter glidende-vindu teknikk. (HDLC er ikke beskrevet i læreboken !)
Nett-laget: dette skal være forbindelses-orientert, eksempelsvis pakke-laget i X.25 (PLP). Tjeneste-repertoaret er i grove trekk beskrevet i oppgave 1.a. Det må kunne etablere forbindelser, sende og motta pakker over denne forbindelsen, og kople ned etter bruk. Benytter glidende-vindu teknikk og flyt-kontroll tilsvarende det vi finner i HDLC.

Adapsjonslaget: dette laget skal tilsynelatende tilby et datagram grensesnitt mot IP-laget og administrere et forbindelses-orientert grensesnitt mot nett-laget. Dette kan tenkes å foregå på følgende måte på sendersiden:

- IP-laget leverer en IP-pakke ned til adapsjonslaget via et SEND.request. Her må IP laget i tillegg levere med nett-adressen til mottaker, på lik linje med IP-over-Ethernet hvor IP-laget leverer med MAC-adressen til mottaker. Neste-hopp adressen i nettet kan være en ruter eller en endemaskin. Sammenhengen mellom destinasjonens nett- og IP-adresse vil i det enkleste tilfelle ligge i en på forhånd konfigurert tabell, som konsulteres etter at fremsendingsbeslutningen er tatt.
- Adapsjonslaget vedlikeholder en tabell over etablerte forbindelser, og vil sjekke om en av disse tilsvarer den oppgitte nett-adressen. Eksisterer forbindelsen, blir IP-pakken sendt over til nett-laget via et N-SEND.request med referanse til forbindelsen. Er der ingen forbindelse, vil adapsjonslaget opprette en ved å benytte en N-Connect.request med oppgitt adresse. Når forbindelsen er opprettet, oppdateres forbindelses-tabellen, og pakken sendes på vanlig måte.
- Vi kan tenke oss at hver forbindelse er assosiert med en ”timer”-verdi. Når forbindelsen brukes, resettes tidsverdien. Dersom timeren går av, slettes forbindelsen ved at adapsjonslaget benytter et N-Disconnect.request kall med referanse til den gitte forbindelsen.

c. *Hvordan kan vi kople sammen IP-laget, som baserer seg på en datagram tjeneste, med det forbindelses-orienterte nettet ? Legg spesiell vekt på å forklare hva slags funksjonalitet vi har behov for.*

Dette er det alt svart på i oppgave 1.b.

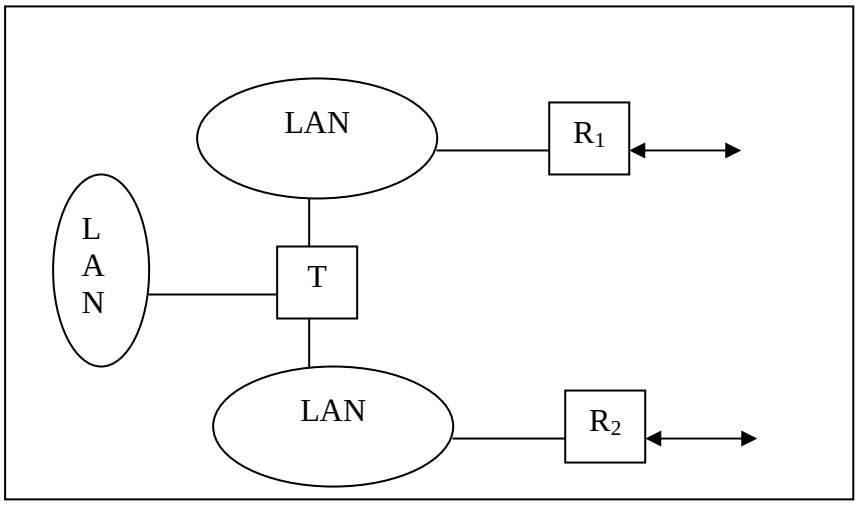
d. *IP-laget og X.25 nettet opererer med ulike adresser. Skaper dette problemer? Hvis svaret er ja, skissere en eller flere alternativer til å løse dette.*

Det finnes ingen enkel transformering mellom IP-adresser og de korresponderende nett-adresser. Her kan det tenkes flere alternative måter å løse dette på. Den enkleste måten er angitt i løsningen på oppgave 1.b, hvor man på forhånd har konfigurert en tabell med denne sammenhengen for alle aktuelle IP-adresser.
Et annet mer dynamisk alternativ vil være en registreringstjeneste. Alle maskiner tilkoplet det forbindelses-orienterte nettet, vil i det de starter opp, registrere seg hos en sentral maskin i nettet. Maskiner kan da etter behov, kontakte denne for å tak i sammenhengen mellom IP-adresser og nett-adresser.

e. *Vis i et diagram sekvensen av operasjoner som må utføres når en ruter vil sende et IP-datagram over X.25 nettet.*

Dette er det vel svart på allerede.

2. Tjenermaskin tilkoplet flere lokalnett



Tjenermaskin T tilkoplet 3 lokalnett

T er en mail-tjener tilkoplet 3 lokalnett som har hver sin adresse-identifikator. Som vist i figuren er to av lokalnettene koplet til det globale internettet via hver sin ruter.

a. Forklar hvordan mailtjeneren adresseres fra en vertsmaskin i det globale internettet og hvordan trafikken fremsendes til mailtjeneren, og hvordan mailtjeneren adresseres fra vertsmaskiner koplet til de lokale nettene.

Hvert grensesnitt til T har en UniK IP-adresse. T kan derfor adresseres på to måter fra det globale internettet og komme inn enten via R_1 eller R_2, avhengig av om IP-adressen til T tilhører det øverste eller nederste LAN i figuren over. IP-adressen til det tredje grensesnittet kan selvsagt ikke benyttes. Det er isolert fra den ytre verden. Vertsmaskiner tilkoplet de tre LAN vil benytte den IP-adressen til T som korresponderer med det LAN de selv sitter på.

b. Anta at trafikken fra det globale internettet inn til mailtjeneren i et gitt øyeblikk fremsendes gjennom ruter R_1. Så går R_1 ned. Får dette konsekvenser for den pågående kommunikasjonen; diskuter dette! Dersom det får konsekvenser, hvordan må vi takle en slik situasjon.

Når R_1 går ned, vil trafikken til T stoppe opp. Trafikken kan ikke omrutes via R2, fordi R2 ikke er veien til T's spesifiserte IP-adresse. Initiativtaker til den berørte kommunikasjonen, må kople ned de ødelagte forbindelsen og så kople opp en ny – men nå med IP-adressen til T på det nedre LAN.

3. Korte spørsmål i forbindelse med linklaget og lokalnett.

Her skal svares med en eller to korte setninger!

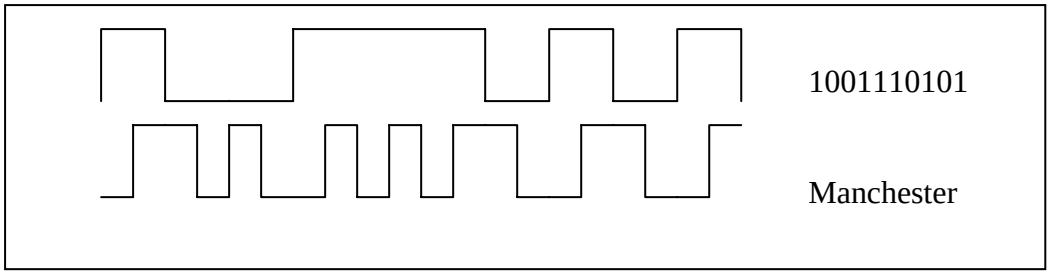
a. Hvilke alternative metoder for flyt kontroll har vi i en punkt-til-punkt link?

Receiver-Ready/Receiver-not-Ready (RR/RNR) i HDCL, glidende vindu, dynamisk justerbart vindu (kreditt-basert) i TCP, blokkerende grensesnitt (grensesnittet mot Ethernet).

b. Hvorfor benytter vi "preamble" i forbindelse med Ethernet?

To årsaker: synkronisering av mottakersiden til senderens bit-takt, og markering av start på Ethernet rammen.

c. Illustrer Manchesterkoding for bit-sekvensen 1001110101



d. Hvorfor benyttes Manchesterkoding, og hvilken konsekvens har den for båndbredden til mediet?

Manchester koding har en transisjon midt i hvert bit-interval. Denne transisjonen benyttes til synkronisering av mottakers klokketakt og fase, og minimaliserer forskyvninger i referansepotensialet på mottakersiden som et resultat av mange nullere eller enere etter hverandre. Manchesterkodingen øker koderaten til det dobbelte av bit-raten, altså 20 Mkodebit/s.

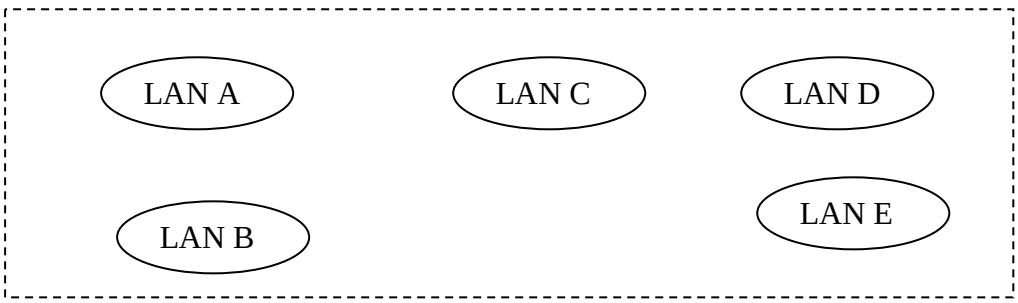
e. Nevn to eller flere fordelaktige egenskaper til et "Token"-ring nett i forhold til et Ethernet.

Betydelig høyere utnyttelsesgrad for et Token-ring nett i forhold til Ethernet. I et Token.ring nett kan vi angi en øvre grense for Token-rotasjonstiden, det vil si på responstidene i nettet. Vi kan benytte prioritet, noe vi ikke kan i et Ethernet, og i tillegg kan sender av en pakke få bekreftet at mottakeren har mottatt pakken, i det senderen fjerner pakken fra ringen.

f. I et Ethernet eller ringnett kan alle maskiner høre alle, og benytter her CSMA teknikk for å kontrollere når man kan sende. Kan vi benytte samme teknikk i et Radio-LAN? Hvis så ikke er tilfelle, forklar hvorfor.

I et radio-LAN må alle sende og motta på samme frekvens, og om flere stasjoner sender samtidig, vil disse ødelegge for hverandre på tilsvarende måte som i et Ethernet. Men en stasjon kan ikke sende og motta samtidig, og vil derfor ikke kunne oppdage kollisjoner. I tillegg kan vi ha "Hidden-terminal" problemet, hvor to stasjoner som er utenfor hverandres rekkevidder sender til en tredje stasjon som kan høre begge. "Carrier-sense" teknikken vil derfor ikke være tilstrekkelig her for å kontrollere hvem som får lov til å sende, men kan selvsagt benyttes og da med mindre effektivitet.

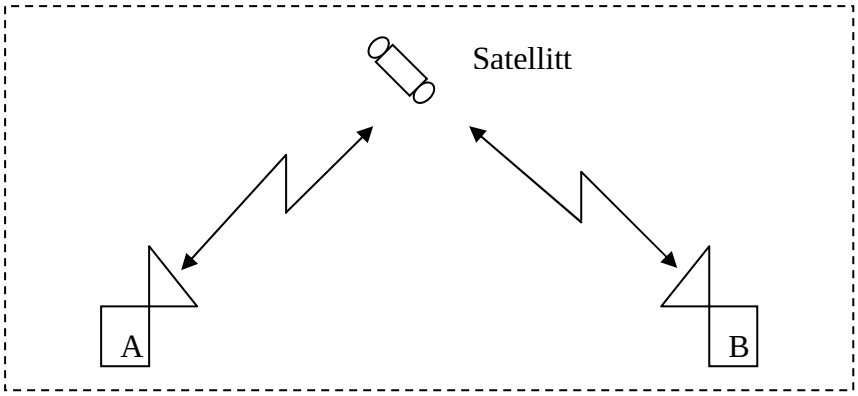
4. Broer og rutere



Som vist i figuren, har vi en samling med Ethernet. Disse ønsker vi å kople sammen. Her har vi ulike alternativer. Diskuter fordeler og ulemper med de ulike alternativene, og gjøre en vurdering av hvilket alternativ du vil velge.

Om vi antar at hvert LAN-segment er 500 meter eller mindre, kan vi kople disse sammen ved hjelp av repeatere, broer, og rutere. Benytter vi repeatere, kan vi kople alle 5 segmentene sammen til et Ethernet. CSMA/CD mekanismen blir da gjennomgående, det vil da si at lokal trafikk innen et segment, interfererer med trafikk på de andre segmentene. Det vil være en felles IP nett-adresse for alle 5 segmentene.
Benytter vi broer for sammenkoplingen, vil CSMA/CD mekanismen bli isolert til hver segment og ikke være gjennomgående. Vi kan da isolere lokal trafikk på hvert segment fra hverandre, og bare trafikk som er ment for maskiner på andre segmenter, vil slippe gjennom de relevante broer. På denne måten får vi bedret utnyttelsesgraden. Det vil være en felles IP nett-adresse for alle 5 segmentene.
Benytter vi rutere for sammenkoplingen, har vi tilsvarende trafikale egenskaper som for broer. Men hvert segment har nå sin egen spesifikke IP nett-adresse.
Hvilke av disse alternativene man vil velge, vil avhenge av trafikkbelastning og trafikkfordeling, av behovet for brannvegger for å hindre trafikk mellom en eller flere segmenter, og om man må være konservativ i bruk av IP-adresser.

5. Transportprotokoll problematikk



To maskiner A og B kommuniserer via satellitt

Vi har en kommunikasjonssituasjon som vist i figuren over, hvor maskinene A og B kommuniserer via en satellittlink med følgende egenskaper: dataraten er 1Mb/s, gangtiden opp til satellitten er 130 mSek, og linken er full dupleks.

A og B ønsker å sender fortløpende datapakker med konstant størrelse på 10.000 bit, som inkluderer pakkehode på 200 bit. Pakkehode inneholder kvitteringer (Ack) på korrekte mottatte datapakker.

a. *På grunn av transmisjonsfeil, vil 1 av 10 pakker bli forkastet på mottakersiden. Hvilken bitfeilsannsynlighet tilsvarer dette?*

Hver pakke er på 10.000 bit og 10 pakker tilsvarer 100.000 bit. Antar vi vilkårlig fordeling av transmisjonsfeil og at kun et bit er korrumpert, blir bitfeil-sannsynligheten lik 10^{-5}.

b. Hvilken pakkestørrelse måtte vi velge om bare 1 av 100 pakker skal bl forkastet på mottakersiden?

Sannsynligheten P for at en pakke med n bit kommer over riktig med en feilrate på $p = 10^{-5}$ er

$$P = (1 - p)^n \sim 1 - n * p$$

For at 1 av 100 pakker feiler, med samme antakelse som over, får vi

$$1 - n * p = 0.99 \rightarrow n = (1 - 0.99)/p = 10^3.$$

Vi må velge en pakkestørrelse på 1000 bit.

c. Hvor stor er netto datarate for de to situasjonene i spørsmål a og b?

Netto datarate i første tilfelle er 1 Mb/s * (10.000 – 200)/10.000 = 0.98 Mb/s
Netto datarate i andre tilfelle er 1 Mb/s * (1000 – 200)/1000 = 0.8 Mb/s

d. Hvor lang er overføringstiden fra det tidspunkt A starter å sende ut en datapakke til den er mottatt av B?

Vi antar her 10.000 bit pr pakke.
Overføringstiden blir = $(10.000/10^6 + 0.130 + 0.130)$sek = 270 millisek.

Vi vil nå spesifisere en enkel pålitelig transportprotokoll som tillater A og B å sende fortløpende og som effektivisere bruken av linken. Det vil si vi vil benytte glidende vindu teknikk.

e. Forklar kort prinsippet for glidende vindu teknikken.

Størrelsen på senderens vindu forteller hvor mange pakker (eller oktetter) senderen kan ha utestående (det vil si sendt) før den må vente på kvittering. Når kvitteringen kommer inn, vil senderens vindu åpnes tilsvarene det antall utestående pakker som blir kvittert. Ved riktig valg av størrelsen på senderens vindu, mottar senderen kvittering på utestående pakker før hele vinduet er fullt. Det vil si at senderen da kan sende fortløpende.

f. Diskuter hvordan du bør velger størrelsen på vinduene på sender og mottakerside og på sekvensnummereringen. (NB! Fortsatt 10.000 bit pakker.)

Først velges senderens vindustørrelse. Det må være lik eller større enn 2 * overføringstiden ganget med dataraten på 1 Mb/s.
D.v.s. senderens vindu W-S må være lik eller større enn 1 Mb/s * 0.540 = 540.000 bit = 54 pakker.
For størst mulig gjennomstrømning, må sender vindu og mottaker vindu W-R være like store.
Hver pakke må sekvensnummereres. Vi trenger et visst antall bit for dette, og som representerer et antall sekvensnummere eller et sekvensnummer rom. Sekvensnummer rommet skal være større eller lik summen av W-S og W-R, i dette tilfelle lik eller større enn 2 * 54 = 108. Det vil si vi må benytte minimum 7 bit for sekvensnummereringen.

g. I de fleste transportprotokoller benyttes "Go-back N", det vil si at ved timeout sendes alt som ligger i sendervinduet om igjen. Diskuter konsekvensene av dette i vårt tilfelle. Kan det tenkes andre alternativer til "Go-back N", og som vil øke effektiviteten i vårt tilfelle? Skisser hvilke endringer i vår protokoll som må gjøres for å kunne benytte alternativet (eller alternativene).

Vi velger W-S lik 64, og ved feil sendes hele vinduet om igjen, selv om det bare er en av 10 pakker som har feilet. Dette er bortkastet kanaltid, øker den gjennomsnittlige overføringstiden og minker den gjennomsnittlige overføringskapasiteten.

For å øke effektiviteten, kunne vi tenke oss en selektiv negativ kvittering fra mottaker på de pakker som feiler, og som sørger for at senderen bare sender om igjen disse pakkene. Det vil si at vi utvider pakkehodet til også å inkludere negative kvitteringer. (Vi vil ikke forlange at studentene skal detaljere dette.)

6. Fjernprosedyrekall (20%)

Du har fått i oppdrag av en forretningsbank å lage en applikasjon der brukerne av banken kan gjøre forespørsler og foreta transaksjoner over Internet. Banken ønsker i første omgang å få implementert følgende funksjoner:

- *saldoforespørsel på et kontonummer som brukeren oppgir*
- *kontoutskrift for siste måned for en konto*
- *giro-innbetaling fra brukerens konto til konti som brukeren oppgir*

Du skal i denne forbindelse løse følgende deloppgaver:

a) Definer de nødvendige datastrukturer som trenges for de ulike oppgavene, og velg en standardisert representasjon (overførings-syntaks) for disse strukturene. Vis hvordan de ferdig kodete strukturene vil se ut, og forklar hvordan de skal brukes i dine løsninger

Det vil selvsagt være mange datastrukturer som passer til dette formålet. Her er det valgt en minimal struktur, som burde være tilstrekkelig. Hvordan man viser strukturen er valgfritt; her er valgt noe a la det som brukes i boka:

1) Saldo-forspørsel:
   Funskjon: **"saldo"**
   Sendte data: char kontonr[7];
   Mottatte data: char saldotekst[30];
   char kontonr[7];
   float belop;

2) Konto-utskrift:
   Funksjon: **"kontoutskr"**
   Sendte data: char kontonr[7];
   Mottatte data: char heading[60];
   char kontonr[7];
   int antall;
   struct linje{
   char dato[4];
   char tekst[30];
   float belop;
   };
   float total;
   Her angir "antall" hvor mange linje-records som inngår i urskriften.

3) Betale giro:
   Funksjon: **"giro"**
   Sendte data: char frakonto[7];
   char tilkonto[7]; /*Forutsettes høyst 7 siffer i kontonummer – blanke til slutt hvis

6.

kortere*/
char mottakernavn[40];
char mottageradr[50];
float belop;
Mottatte data: bool OK;

Det forutsettes at klient programvaren hjelper brukeren med å taste inn de nødvendige dataelementene. Her bryr vi oss bare om det som skal leveres til tjenermaskinen ved utførelsen av fjernprosedyrekallene.

Selve kodingen av data-elementene kan f.eks. foretas i XDR (ASN.1 eller NDR).
Nedenfor er gitt et eksempel på kontonummer kodet i XDR. De andre data-elementene kodes på tilsvarende måte (orker ikke å lage alle sammen).

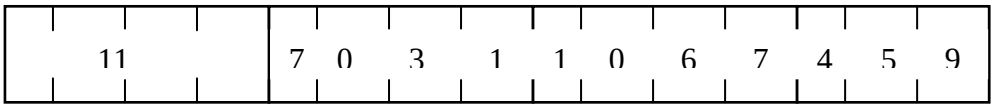


Fig.1: Eksempel på koding

b) De tre funksjonene ovenfor skal utføres på en tjenermaskin som har tilgang til bankens databaser. Fjernprosedyrekall (RPC) fra brukernes datamaskiner skal benyttes for å aktivere de ulike funksjonene. Forklar hvordan klientmaskinene finner fram til den riktige funksjonen, og identifiser hvilken prosess i maskinen som håndterer dette.

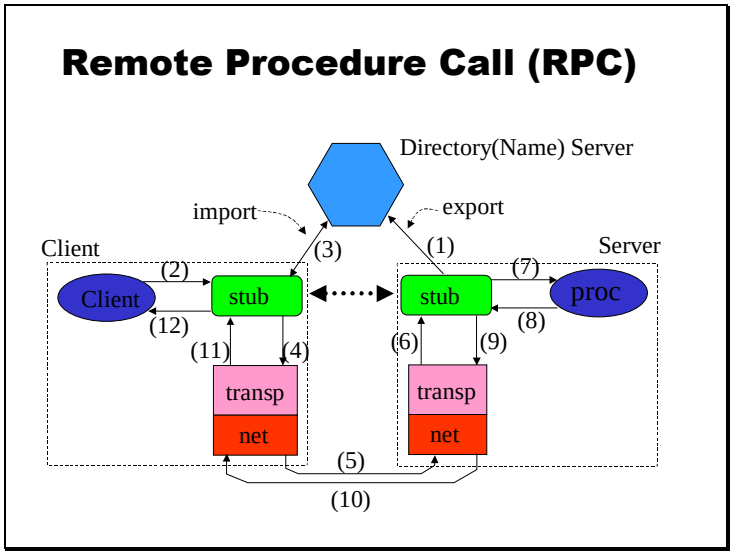


Fig.2: Fjernprosedyrekall

Det dreier seg her om kallene på de prosedyrene jeg har kalt **"saldo"**, **"kontoutskr"** og **"giro**" i oppgave a). Her er vi ute etter at studenten skal forklare bruken av navne-tjeneren på fig.2; dvs. at

tjener-maskinen eksporterer alle sine prosedyrer som det er mulig å kalle fra en klient til en navne-tjener, mao. de tre ovenfor nevnte for vårt formål. Når klient stub´en får beskjed om å utføre et kall på vegne av klienten, vil den se om den finner navnet på den fjerne prosedyren i navnetjeneren. Hvis så er tilfelle, vil den f.eks. bruke grensesnitt-beskrivelsen for prosedyra og en stub-compiler til å generere klient og tjener stub´ene. Andre måter å generere stub´ene på er også beskrevet i boka, og bør godtas.

c) Kall på de ulike funksjonene innebærer til dels overføring av sensitive data over Internet. Hva ville du gjøre for å forhindre innsyn i disse dataene.

Vi er her ute etter litt vurderinger vedrørende sikkerhet fra studenten, og gode resonementer bør premieres mer enn detaljert referat fra læreboka.

Følgende forhold anses relevante her:

- For i noen grad å kunne forsikre seg om at brukeren virkelig er den han/hun gir seg ut for, bør det kanskje kreves at bruker logger seg på tjener-maskinen før det gis tilgang til de ulike prosedyrekallene. Passord bør krypteres. En kan kanskje gå ut ifra at dette er tilstrekkelig sikkerhet til at brukeren kan få se saldo og evt. bestille en konto-utskrift.
- Noe annet er det når brukeren begynner å ta ut penger fra kontoen sin! Alle liknende (bank-)systemer avkrever derfor en eller annen form for bekreftelse fra brukeren om at de oppgitte ttransaksjonene virkelig skal utføres før de endrer databasen (og manipulerer på brukerens konto). Dette bør være en mest mulig sikker bekreftelse, f.eks. ved å benytte en kode-generator hjemme hos brukeren, som generer ny kode hver gang transaksjoner skal bekreftes. Koden bør i tillegg gå kryptert.
- Endelig kunne det være ønskelig å sikre alle data som sendes over nettet for derved å hindre innsyn, modifisering mv. Dette innebærer bruk av kryptering for alle data som sendes, f.eks. ved å ta i bruk SSL, evt. med sanntids forhandling, sertifikater mv.

d) Hvilke (semantiske) krav må stilles til selve utførelsen av de ulike funksjons-kallene på tjenermaskinen?

Dette går på hva tjeneren må kunne garantere når det gjelder utførelsen av de fjerne prosedyrekallene. Når det gjelder kallene på **saldo** og **kontoutskr** vil man kunne leve med minst-en-gang (at-least-once) semantikk siden kallene vanligvis ikke manipulerer på databasen, men bare leverer fra seg info om det som ligger der. Klienten må eventuelt kunne ordne opp i duplikatmeldinger på en passende måte.

Når det gjelder kallene på **giro** rutinen, er det imidlertid et absolutt krav at tjeneren kan garantere høyst-en-gang (at-most-once) semantikk siden det her manipuleres på databasen. Helst burde man tilby nøyaktig-en-gang (exactly-once) semantikk. Dersom kun førstnevnte variant kan garanteres, må det finnes andre mekanismer i banksystemet som informerer brukeren om hva som egentlig er status for kontoen og utførelsen av de ønskede transaksjonene (lite ønskelig).