Løsningsforslag INF240 våren 2003

1. **Referansmodellen;** begreper og betydning.
   a. Utdyp betydningen av tjenestegrensesnitt og protokollgrensesnitt.

Tjenestegrensesnittet beskriver tjenesten et lag tilbyr til laget over. Så lenge den semantiske betydningen bibeholdes, kan tjenestegrensesnittet i en maskin implementeres på en måte, mens det tilsvarende grensesnittet i en annen maskin kan implementeres på en annen måte. Videre kan vi, ut fra beskrivelsen av tjeneste-grensesnittet, ikke trekke noen definitive konklusjoner om hvordan tjenesten blir realisert i det tjenesteytende laget, eller i det og i kombinasjon med lagene under. Et lag kan derfor godt tilby en forbindelsesorientert tjeneste til laget over, men selv bygge på en forbindelsesfri eller forbindelsesorientert tjeneste fra laget under. Eks. TCP over IP og TP0 over PLP-laget i X.25.
Protokollgrensesnittet definerer så vel et semantisk som syntaktisk (PDU-formatene) grensesnitt, og må være identisk i de maskiner som ønsker å kommunisere med hverandre. Protokollen selv er mekanismen entiteter på samme nivå benytter for å samarbeide, for derved å kunne tilby en gitt tjeneste til laget over.

   b. Hvilken betydning har et ”Service Access Point” (SAP) og hvordan relateres det til egenskaper i internett protokoll arkitekturen?

En **SAP** er et adresserbart referansepunkt mellom to lag og knytter en assosiasjon mellom en entitet i et lag og en entitiet i laget over eller under. En SAP er alltid assosiert med tjenestesiden av et lag, og et lags tjeneste tilbys alltid via en gitt SAP. En SAP kan realiseres som en dupleks køstruktur mellom to lag. I internett arkitekturen finner vi tilsvarende begreper, men med andre navn. TCP tilbyr sine tjenester via TCP-porter og UDP via **UDP-porter**. IP protokollen inneholder et felt i IP-hodet kalt ”**Protocol**”. Det peker ut protokoll-entiteten i laget over – TCP, UDP, etc), og er derfor identisk med en SAP.

   c. Hva er betydningen av et samtaleunivers. Illustrer og beskriv dette gjennom et eksempel.

Et **samtaleunivers** beskriver den felles oppfatning to kommuniserende (samarbeidende) applikasjonsentiteter må ha for at de skal kunne samarbeide på en konstruktiv måte. I kurset har vi benyttet et eksempel med et flyselskap og et reisebyrå. Denne felles forståelsen kan vi uttrykke i et sett av prosedyrer og beskrive disse i en egnet syntaktisk form, for eks basert på ASN.1. Denne beskrivelsen kan så konverteres (kompileres) over i C, C++, eller andre språk, avhengig av hvilket programmeringsspråk vi ønsker å benytte for vår applikasjonsutvikling. Så lenge den semantiske betydningen ikke endres, kan samtaleuniverset ha forskjellig representasjon i forskjellige maskiner.

   d. Hva forstår du med overføringssyntaks? Hvordan er dette tatt hånd om i ISO’s referansemodell og i internett-modellen?

Ulike hardware, ulike operativsystemer, og ulike programmeringsspråk tilsier at det ikke er gitt at informasjon som overføres mellom applikasjonsprosesser i to vilkårlige maskiner blir oppfattet riktig, selv om de to applikasjonsprosessene har samme forståelse av et felles samtaleunivers. Det vil si at en applikasjonsprosess presenterer sin informasjon på sin ”naturlige” syntaktiske form, mens den andre prosessen ville ha benyttet en annen form. Det medfører at applikasjonsprosessene ikke forstår hverandres budskap. Informasjonen som overføres mellom applikasjonsprosessene må derfor konverteres over i en felles syntaktisk form, kalt overføringssyntaks, før den sendes ut. Mottakersiden konverterer så fra overføringssyntaks til den lokal syntaks som er egnet på denne siden. I ISO’s referansemodell utføres denne konverteringen av presentasjonslaget. Her var det også lagt opp til at man skulle kunne forhandle om hvilken overføringssyntaks man ville benytte.

Internett modellen inneholder intet presentasjonslag, så det er applikasjonsprosessenes ansvar å foreta denne konverteringen selv.

e. ISO's referansemodell fremstilles ofte som en 7-lags modell mens internett modellen fremstilles med 4 lag. Betyr dette at de to modellene ikke harmonerer med hverandre? Diskuter dette!

ISO's referansemodell ble utviklet før internett teknologien var moden nok og akseptert. Ser vi nøye på modellen, med lokalnett og internett øyne, har vi i virkeligheten 10 lag: fysisk, medium aksess, logisk link, nettlag (subnett laget), nett-adapsjons lag, internett lag, sesjonslag, presentasjonslag, og applikasjonslag. Applikasjonene kunne så benytte de mekanismer i applikasjonslaget som applikasjonene hadde bruk for. I internett modellen har man selvsagt bruk for den samme totale funksjonaliteten. Men her har man abstrahert vekk uvesentlige ting, sett med internett øyne. Det vil si at mindre viktig lag har blitt slått sammen, for eks at lagene: fysisk, link, subnett og mulig adapsjonslag sammenfattes i et "vertsmaskin-til-nett" lag eller "Network Driver". Så har vi internett laget, transport laget, og på toppen applikasjonslaget. Sesjonslaget har man ikke sett noen nytte av. Og presentasjonslags og applikasjonslags funksjoner bygges inn etter behov i selve applikasjonene. Det er derfor ikke noe motsetningsforhold mellom ISO's referansemodell og internett modellen.

2. Unix sockets

a. Forklar kort den funksjonelle betydningen av begrepet "Sockets" og hvordan sockets relateres til begreper i Referansemodellen.

En **socket** er et referansepunkt mellom en applikasjonsprosess og en transportprotokoll. En socket kan grovt sett tenkes som er dupleks køstruktur på grenseflaten mellom transportlaget og applikasjonsprosessen og en tilordnet datastruktur, kalt Transport Kontroll Blokk (TCB). TCB vil inneholde all tilstandsinformasjon for en gitt forbindelse, og vil bestå av to deler. Den ene delen vil inneholde alle attributter og tilstandsinformasjon for lokalsiden. Den andre delen vil inneholde tilsvarende informasjon for den andre siden.
Applikasjonsprosessen må be om å få utlevert en socket, via et socket systemkall, og må oppgi om socket-en skal være assosiert med TCP, UDP, eller annen aktuell transportprotokoll. Socket-kallet returnerer med en referanse til en socket (**Socket-ID**), og den tilhørende transportprotokoll **porten** (SAP) vil befinne seg som en attributt i TCB. Selve socket-en kan derfor ses på som realiseringen av en SAP, mens Socket-ID og porten kan ses på som referanser til denne strukturen, henholdsvis sett fra applikasjonens side og fra transportlagets side.

b. I kommunikasjonssammenheng gjør vi ofte et skille mellom de to sidene i en kommunikasjon. Hva er betydningen av dette skillet?

Den ene siden i en gitt kommunikasjon kalles **klienten**, og er den siden som aktiv og tar initiativet til kommunikasjonen. Den andre siden, kalt **tjeneren**, er reaktiv, og prepararer seg for mulige kommunikasjons initiativ. Det vil si tjeneren bringer seg selv i en lyttende tilstand, og derved gjøre seg mottakelig for oppkall. Tjenersiden vil normalt representere en gitt tjeneste. Hver tjeneste er assosiert med en globalt kjent port. Tjenersiden vil derfor lytte på sin spesifikke port, spesifisert et systemkall, kalt Bind. Dette reflekterer seg i TCB, hvor bare den lokale siden (Tjenersiden) inneholder informasjon og som identifiserer dette endepunktet for en mulig forbindelse. Klientsiden får utlevert en port, større enn 1023, når klienten ønsker å sette opp en forbindelse.

c. Gi en funksjonell beskrivelse av socket-kallet Connect! Hva er forutsetningen for at et Connect-kall aksepteres lokalt? Vi forutsetter at vi ønsker å benytte TCP.

Forutsetningen for at Connect-kallet skal aksepteres på lokal side (klientsiden) et at applikasjonsprosessen på forhånd har fått utlevert en TCP-socket. De attributter som må leveres med i Connect-kaller er: IP-adressen og port-id for mottakersiden. Denne informasjonen lagres i TCB-strukturen på klientsiden, sammen med endepunkt-beskrivelsen for klienten. Deretter starter tre-veis håndtrykket: SYN, SYN-ACK, og ACK.

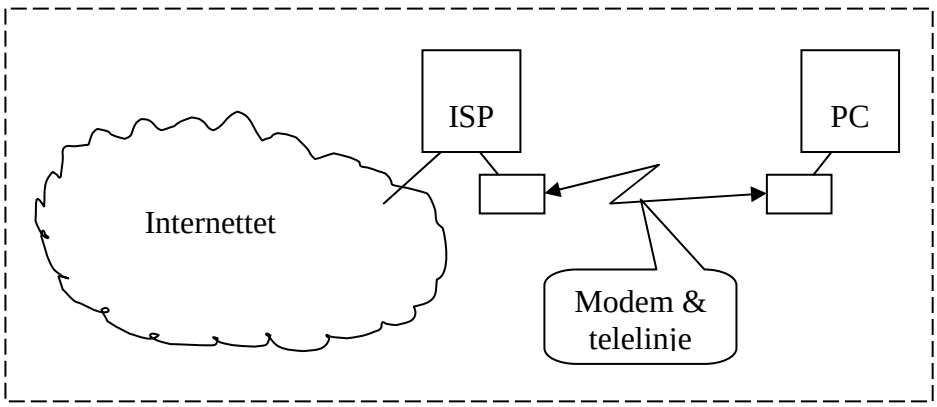
d. Hvilke parametere inngår i kallet og hva er betingelsen for at et Connect-kall resulterer i etablering av en forbindelse? Hva skjer lokalt på begge sider og hva utveksles over nettet? Diskuter dette, og illustrer gjerne dette ved hjelp av tilstands-maskin modeller for de to sidene!

Vi har alt nevnt de parametrene som må inngå i Connect-kallet. På klientsiden vil Connect-kallet føre til at begge deler av TCB strukturen fylles inn, det vil si at den identifiserer begge endepunkter for en forbindelse. Betingelsen for at utførelsen av kallet skal resultere i etableringen av en forbindelse, er at tjenersiden er i lyttende tilstand og ikke har brukt opp sin kvote; det vil si hvor mange parallelle oppkall tjenersiden er villig til å akseptere (spesifisert i Listen kallet). Kliensiden går fra Lukket til SYN-sent tilstand i det tre-veis håndtrykket starter, og går over i Etablert tilstand når SYN-ACK mottas. Tilstanden forbindelsen befinner seg, reflekteres i TCB strukturen. Tjenersiden går fra Listen-tilstand til SYN-mottatt når SYN pakken mottas, og så over i Etablert tilstand når ACK mottas. Også her reflekteres tilstanden tjenersiden befinner seg i, i TCB strukturen på tjenersiden. Når SYN-pakken mottas, blir klientsiden i TCB strukturen fylt inn, slik at begge endepunkter for forbindelsen også er identifisert på Tjenersiden.

e. Vi antar at det er etablert mange forbindelser fra vår maskin og ut til mange andre. Hva er fremgangsmåten for ulike applikasjoner i vår maskin med hensyn til å sende data ut på riktig forbindelse? Og hvordan er fremgangsmåten i prosesseringen av innkomne datapakker med hensyn til å få riktig pakke inn på riktig forbindelse?

Hver forbindelse ut fra vår maskin er assosiert med en gitt applikasjonsprosess og en gitt socket struktur. Applikasjonsprosessen refererer så til denne når en pakke skal sendes, via den tilhørende **socket-ID**. Når en IP-pakke mottas, vil innholdet av IP-pakken, sammen med pseudohodet overføres til transportlaget (vi antar TCP). Avsenders IP-adresse og port-ID, samt mottakersidens port-ID (muligens inkludert mottakers IP-adresse) benyttes så i et søk gjennom listen av TCBer, til match finnes. Deretter prosesseres TCP-pakken, med referanse til riktige TCB-struktur, og som forhåpentligvis ender opp med at innholdet av TCP-pakken settes inn i kø-strukturen til riktige applikasjonsprosess.

3. Funksjonell arkitektur i oppringt tilgang til internettet.



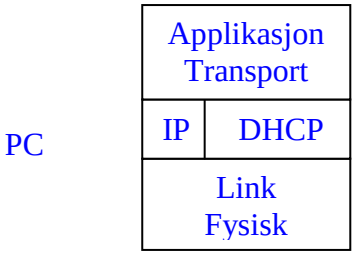
Figuren illustrerer enhetene som inngår i et oppringt samband som kopler din hjemme-PC til internettet via telefon og modem og en maskin hos en ISP (Internet Service Provider), hvor vi har et

abonnement. ISPen kan betjene mange oppringte samband samtidig. I denne oppkoplingen skal din hjemme-PC opptre som en normal verstmaskin i internettet.

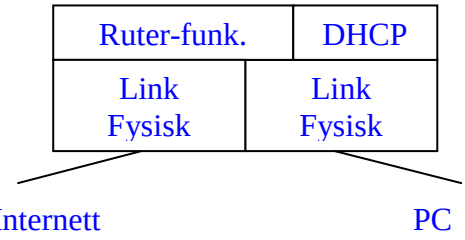
a. Med utgangspunkt i figuren over, illustrere og beskriv protokollstrukturen i ISP-maskinen og i hjemme-PCen.

Vi forutsetter at modemforbindelsen, når den er etablert, er funksjonelt identisk med en vanlig fysisk datalinje. Over denne linjen kan vi benytte en egnet link protokoll, for eks PPP. Vi forutsetter at PCen får utlevert en IP adresse i det modemforbindelsen etableres, via DHCP protokollen.

Hjemme-PC-en vil ha standard protokollstruktur, omtrent som vist i figuren til høyre.



ISP maskinen kan tenkes å ha følgende struktur



Ruter-funksjonen vil, via en fremsendingstabell, holde greie på hvilket grensesnitt IP-adressen til PC-en befinner seg på.

b. Hvordan holder ISP-maskinen orden på et sett av samtidig oppringte samband, slik at den sender datapakker på det riktige modemsamband. Illustrer og beskriv.

Hvert oppringt samband assosieres med et fysisk modem-grensesnitt og når DHCP har gjort jobben sin, med en gitt utlevert IP-adresse. Grensesnitt-ID og tilhørende IP-adresse registreres i fremsendingstabellen, slik at pakker automatisk fremsendes på riktig grensesnitt.

c. Anta at du har flere liknende arbeidsplasser som du benytter på denne måten. Du benytter den samme PC på disse plassene. Har dette noen innvirking på internett adressen til PCen? Diskutere dette.

Når PC-en tilkoples et gitt arbeidssted, utleveres en dynamisk IP-adresse. Dette gjør det svært vanskelig for andre å adressere PC-en, for eks å sende epost til denne. I en slik operasjonsmodus vil det være nødvendig at postkassen ligge på en fast maskin med fast adresse, og at det er mail klienten som ligger i PCen.

Vi kan også tenke oss et opplegg a la Mobil-IP, men det vil være vesentlig mer komplisert å realisere enn det vi har skissert.

4. Fjernprosedyrekall (RPC)

a. Beskriv elementene som inngår i et RPC system og diskuter deres funksjonelle oppgaver. Dette skulle være nokså rett frem. Studentene har foiler som illustrerer

dette. Vi kan benytte TCP eller UDP i vår løsning. Hvordan vil dette influere på funksjonaliteten i de elementene som inngår, dersom vi vil håndtere sporadiske transmisjonsfeil? For å gjøre det enklere, antar vi at en RPC-transaksjon består i en kort melding (pakke) i hver retning.

Vi kan benytte TCP eller UDP. Sporadiske feil håndteres godt av TCP, men siden vi benytter en-pakke meldinger, gir TCP en betydelig overhead i opp og nedkopling av forbindelser. Her kan vi med fordel benytte UDP, og bygge inn påliteligheten i applikasjonsprosessene. Vi må benytte sjekksumtest, for å detektere mulige transmisjonsfeil. Klienten setter en timerverdi i det pakken sendes. Kommer det ikke noe svar innen utløpet av ventetiden, sendes pakken på nytt. Dette gjøres et visst antall ganger før man gir opp. Vi må også benytte sekvensno i våre pakker, slik at man kan skille mellom nye og gamle instanser av slike transaksjoner. Dette er en grei fremgangsmåte for klientsiden. Tjenersiden må også ha muligheten til å få beskjed om at resultatet som returneres klient virkelig er kommet frem, for eks ved bruk av eksplisitt kvittering, parret med timeout og retransmisjon. En annen strategi, er å la klienten spørre, selv om tjenersiden tidligere har returnert et svar, men som er blitt ødelagt på veien. Denne gjentatte spørringen, må resultere i at tjeneren ignorerer spørsmålet, men sender svaret nok en gang

b. Hvilke argumenter vil du benytte for velge enten TCP eller UDP?

Dette er vel i hovedsak svart på i spørsmålet over

c. Det er alltid muligheter for alvorlige feil, som brudd på kommunikasjonslinjer eller systemkrasj. I slike tilfeller må en RPC-transaksjon, som ikke lot seg gjennomføre fullt og helt, gjentas når nett og endesystemer igjen er operative. Anta at initiativtakeren til en RPC transaksjon, ber om endringer i for eksempel en bankkonto. Svaret som returneres, gir status etter at endringene er utført. Hvordan vil slike alvorlige feil influere på resultatet av en RPC transaksjon, og diskuter hvordan vi kan takle de ulike kategorier av feil?

RPC-transaksjoner som medfører tilstandsendringer, må takles slik at transaksjon bare blir utføret en gang. Dette medfører at begge sider i en slik transaksjon må registrere hvor de til enhver tid befinner seg i gjennomføringen av transaksjonen, slik at de, etter at systemene igjen er operative, kan ta fatt der de slapp. Slik tilstandsinformasjon må lagres slik at de ikke ødelegges ved systemkrasj.

5. Mobil kommunikasjon (25%).

Du skal besvare følgende spørsmål:

a) Beskriv hvordan du vil realisere denne applikasjonen, og gjør spesielt rede for:
- hva slags nett som benyttes mellom de ulike utstyrsenhetene; begrunn svaret.

Tanken her er at kunne bruke Bluetooth mellom utstyrsenhetene i sykebilen. Mellom Lap-topen i bilen og sykehuset benyttes wireless MAN (802.16); det forutsettes her at sykehuset har en ruter som kan videresende kall til akutt-mottaket og andre stasjoner internt. Vakthavende leges Lap-top knyttes til et trådløst Ethernet (802.11), som forutsettes å kunne nås overalt internt på sykehuset. Både maskinen ved akutt-mottaket og Lap-topen til vakthavende må kunne adresseres via internet-adresser.

- hvilke nett-forbindelser som eventuelt må opprettes, og hvilke meldinger som utveksles mellom de ulike utstyrsenhetene (i form av metodekall)

Følgende oppkoplinger må gjøres (her er vi åpne for ulike løsningsforslag):
Måleutstyret og GPS knyttes til Lap-top i sykebilen via SCO linker, og Lap-topen henter data fra dette utstyret. Måleutstyret vil sannsynligvis levere data kontinuerlig, mens Lap-topen f.eks. henter ut koordinater hvert 10 sek. Fra GPS-en.
Lap-topen kopler seg til akutt-mottaket og legens Lap-top. Dette kan tenkes gjort på flere måter; hvis multicast er tilgjengelig, opprettes to forbindelser til både akutt-mottaket og vakthavenes Lap-top. Ellers kan det opprettes fire separate forbindelser. En kan også tenke seg at akutt-mottakes maskin releer videre til vakthavendes maskin. Poenget er at det etter oppkopling finnes en forbindelse til akutt-mottaket og legen, som benyttes for kontinuerlig overføring av medisinske data, og at det finnes en tilsvarende forbindelse som benyttes for overføring av de geografiske koordinatene til sykebilen. Det er ikke meningen at studenten skal gå inn på hvordan disse dataene er formatert og displayes på de ulike PC-ene.
Eksempler på typer av meldinger vil da kunne være (etter oppkopling):

start-medidata-sending	(det forutsettes her at det er kjent for utstyret hvilke data som sendes)
end-mididata-sending	(denne vil kunne aktivires både fra sykebilen og legen; legen vil så kunne sende tekstlig råd om behandling, og evt. skru på sendingen av medisinske data igjen ved å kalle start-medidata-sending)
start-tekst	
end-tekst	
start-GPS(periode)	
send-GPS(lengde, bredde)	
end-GPS	
hent-pasientinfo(id)	(Sendes til RIT via jordbunden link fra vakthavendes Lap-top)

*****Vi bør være fleksible ift. ulike forslag her*****

b) Hvis pasientens identitet foreligger, vil legen kunne hente ut vedkommendes journal fra RIT dersom denne foreligger. Dette gjøres ved fjernprosedyre-kall (RPC). Gjør rede for hva slags kall-semantikk du ville velge for slike kall; begrunn svaret.
Dette RPC kallet endrer ikke noe i databasen ved RIT, slik at ”at least once” kall-semantikk brukes. Mottakerens Lap-top kan evt. fikse duplikater.

c) Er det noen overføringer som har spesielle behov for å sikres mot innsyn? Hvordan kan dette i så fall ivaretas?
Poenget her er at studenten skal se at pasient-data er særdeles følsomme data, slik at det er linken til RIT som er sårbar. All info som flyttes over denne linken må derfor sikres med en meget pålitelig kryptering.

6. Flervalgspørsmål (5%).
Her kreves 100% riktige svar for å få 1.0.
30% riktig gir 4.0.
Noenlunde lineært imellom disse ytterpunktene.

a) Et kabel-brudd i en buss topologi stopper all overføring.
   1: maskenett
   2: buss
   3: stjernenett
   4: tre-struktur

b) Node-til-node overføring av data-enheter er ansvaret til data link laget.
   1: fysisk

2: data link
3: transport
4: nett

c) Modulasjon av et analogt signal kan oppnås ved modulasjon av pkt.4 til bære-bølgen.
1: amplitude
2: frekvens
3: fasen
4: hvilken som helst av de ovenforstående

d) Ved asynkron overføring er tids-gapet mellom bytes variabelt.
1: fast
2: variabelt
3: en funksjon av data-raten
4: null

e) I en omgivelse med mange høy-spennings innretninger, vil det beste overføringsmediet være optisk fiber.
1: tvunnet par kabel
2: koaksial-kabel
3: optisk fiber
4: atmosfæren

f) Feil-deteksjon gjøres vanligvis i data link laget i OSI-modellen.
1: fysiske
2: data link
3: nettlaget
4: hvilket som helst av de ovenforstående

g) Flytkontroll er nødvendig for å unngå overflyt i mottaker-buffer.
1: bit-feil
2: overflyt i sender-bufferet
3: overflyt i mottaker-buffer
4: kollisjon mellom sender og mottaker

h) HDLC er en bit-orintert protokoll.
1: tegn-orientert
2: bit-orientert
3: byte-orientert
4: teller-orientert

i) Hvilken type svitsjer bruker hele kapasiteten til en dedisert link?
1: linje-svitsjing
2: datagram pakkesvitsjing
3: virtuell krets (VC; Virtual Circuit) pakkesvitsjing
4: meldings-svitsjing

j) Hvilken av følgende enheter bruker størst antall lag i OSI-modellen?
1: bro
2: repeater
3: ruter
4: gateway