Løsningsforslag eksamen TTM4100 14. mai 2020

NB! Forslaget til poenggivning er veiledende, ikke absolutt. Hvis en student har gitt et omfattende og grundig svar, men glemt et av de momentene vi forventet, kan vedkommende likevel gies full score på oppgaven. Det kan også hende de har gode tilleggsmomenter som jeg mangler i løsningsforslaget.

1.1 UDP versus TCP

Gjør rede for forskjellene i funksjonalitet mellom de to transportlags-protokollene UDP ("User Datagram Protocol") og TCP ("Transmission Control Protocol"). (6 p).

Løsningsforslag (med forslag til poenggivning):

UDP er en ekte datagram-protokoll siden hver pakke sendes uavhengig av andre pakker til en mottaker; dvs. det settes **ikke** opp en forbindelse først. (1 poeng)

TCP er en forbindelsesorientert protokoll, hvor en forbindelse må settes opp via «three-way handshake» før nyttedata kan overføres. (1 poeng)

TCP har implementert både flytkontroll («flow control») og overlastkontroll («congestion control»). (1 poeng)

UDP har verken flytkontroll eller overlastkontroll. (1 poeng)

Både UDP og TCP har implementert en sjekksum («Checksum») i pakkehodet, for å detektere feil i data som overføres. (1 poeng)

UDP pakkehodet er kun 8 oktetter («bytes») som inneholder portadresser, lengde og sjekksum. TCP trenger 20 oktetter bl.a. for å i tillegg holde rede på informasjon som trengs i forbindelse med oppsett av forbindelsen og implementasjon av flytkontroll. (1 poeng)

1.2 Flytkontroll og Overlastkontroll / Flow control and congestion control

Gjør rede for de to begrepene flytkontroll ("Flow control") og overlastkontroll ("Congestion control") i ett nett. (Stikkord: Hva brukes de for; hvilke enheter i nettet er involvert; hva er forskjellen i funksjonalitet for de to mekanismene). (7 p).

Løsningsforslag (med forslag til poenggivning):

Hovedhensikten med flytkontroll er å beskytte mottaker mot å få informasjon raskere enn det som kan håndteres. (1 poeng)

Hovedhensikten med overlastkontroll er å (helst) unngå at nettet overbelastes, eller komme seg tilbake til normal situasjon hvis det skjer. (1 poeng)

Kun sender og mottaker er involvert i implementasjon av flytkontroll ved at mottaker må gi beskjed til sender om å senke hastigheten hvis mottakers buffer går fulle. (2 poeng)

For overlastkontroll kan også nettet selv (f.eks. svitsjer/rutere) i noen type systemer være involvert og signalere eksplisitt overbelastning, men det er ikke tilfelle for TCP

hvor også denne må håndteres mellom sender og mottaker (ende-til-ende på transportlaget). For TCP justeres senderater basert på observerbare konsekvenser av overbelastning i nettet: som at flere kvitteringer uteblir eller gjentas (p.g.a. pakketap hvis helt fulle buffere i nettet) eller tar for lang tid (p.g.a. fullere buffere/lengre køer i nettet). (3 poeng)

1.3 Flytkontroll i TCP / Flow control in TCP

Gjør rede for hvordan flytkontroll er implementert i TCP protokollen. (Stikkord: virkemåte; hvilke enheter i nettet deltar; hvilke funksjoner ivaretas av de ulike enhetene). (7 p).

Løsningsforslag (med forslag til poenggivning):

Kun sender og mottaker er involvert i flytkontroll i TCP (på transportlaget). (1 poeng)

Sender vedlikeholder et variabelt mottakervindu («receive window») som prøver å holde rede på hvor mye ledig buffer det er hos mottaker. (2 poeng)

Denne justeres basert på det som sendes og kvitteringer som mottas fra mottaker. Disse kvitteringene sier hvor mange oktetter som er vellykket mottatt. Hvis mottakervinduet får for lav verdi må sender vente med å sende til flere kvitteringer mottas. (2 poeng)

Kvitteringer er kumulative i TCP, så hvert segment behøver ikke egen kvittering. (1 poeng)

Sender vil sende segmenter på ny hvis ikke kvittering mottas innen en viss tid («timeout»). (1 poeng)

2.1 Feildeteksjon og korreksjon / Error detection and correction

Gi en oversikt over feil-deteksjons og -korreksjons mekanismene som er dekket av pensum. Forklar med egne ord virkemåten til hver av dem. (7 p).

Løsningsforslag (med forslag til poenggivning):

(Kanskje en litt «løs» oppgave siden det finnes mange mekanismer i pensum. Jeg tenkte nok mest på de som er bekrevet i kapittel 6.2. Det som omtales andre steder i pensum er vel i hovedsak anvendelse av disse prinsippene).

(Enklest mulig) Paritetssjekk: En legger til ett ekstra bit etter en bitsekvens for å detektere om det har skjedd feil i overføringen. Ved lik («even») paritet vil en føye til en 1 hvis antall enere i bitstrengen er ulik («odd»), eller 0 hvis den allerede er lik. Mottaker vet da at «noe» er galt hvis antall enere er ulikt ved mottak. Hvis mediet det overføres over har svært lav bitfeilrate kan dette være bra nok, ellers ikke. (1 poeng)

2-dimensjonal paritetssjekk: De samme bit-ene inngår nå i paritetssjekk i to dimensjoner. Dette øker påliteligheten til mekanismen og gjør det også mulig å

korrigere enkeltbitfeil. To samtidige bitfeil kan også detekteres. Men fremdeles mest egnet for medier med svært lav bitfeilrate. (2 poeng)

Sjekksum metoder: Ofte brukt i hodet på protokoller (f.eks. UDP og TCP; «Internet checksum») for å detektere feil i data (og i noen grad hodet selv). Generelt går disse ut på å summere sammen mange tall, men med «wrap-around» av overflyt (en slags modulo beregning) slik at en alltid får en sjekksum med fast lengde som sendes med segmentet til mottaker. En lignende operasjon gjøres da hos mottaker for å sjekke at det ikke har blitt feil i overføringen. Lite ressurskrevende, men også mye svakere til å detektere feil enn f.eks. CRC (se nedenfor). (1 poeng)

Cyclic Redundancy Check (CRC): En type sjekksum metode der sjekksummen finnes som resten fra en spesiell type divisjon. Data som skal sendes divideres med en kjent binær verdi med lengde r + 1 (må alltid starte med en «1» bit) som kalles generator. Denne er kjent for og brukes av både sender og mottaker. Data pluss r nuller divideres med generatoren. Alle beregninger er basert på modulo-2 aritmetikk uten låning eller overflyt, dvs. i praksis XOR operasjoner på bitnivå. Resten en ender opp med legges da til de opprinnelige dataene (i stedet for de r nullene som ble lagt til før divisjonen) og overføres til mottaker. Når mottaker gjør samme divisjon skal da resten bli null hvis feilfritt. (Kanskje litt utenfor pensum, men grunnen til at det gjøres på denne måten er at dette er vellegnet for å bli implementert i hardware som skiftregistre, for effektiv beregning på linklaget). (3 poeng)

2.2 CSMA/CD og CSMA/CA / CSMA/CD and CSMA/CA

Gjør rede for medium aksess protokollene CSMA/CD og CSMA/CA. (Stikkord: virkemåte; hvordan de er ulike; hva de typisk brukes til). (7 p).

Løsningsforslag (med forslag til poenggivning):

CS = “Carrier Sense”: Indikerer at en lytter på mediet og ikke sender når en detekterer aktivitet. (0,5 poeng)

MA = “Multiple access”: Flere har aksess til mediet uavhengig av hverandre og det er ingen koordinering mellom dem. (0,5 poeng)

CD = “Collision Detect”: Indikerer at en kan oppdage om en har kollidert etter at en har sendt noe ut på mediet. (0,5 poeng)

CA = “Collision Avoidance”: Indikerer at en forsøker (i større grad) å unngå å kollidere med andre via den mekanismen som implementeres. (0,5 poeng)

I CSMA/CD kan en starte å sende straks, hvis en detekterer at mediet er ledig. Hvis kollisjon detekteres vil en trekke tid til neste forsøk og prøve igjen. Tiden en trekker øker som funksjon av antall ganger en har kollidert. (1 poeng)

I CSMA/CA implementeres en mer konservativ aksessmetode der en uansett må trekke en forsinkelse til en får lov til å sende, som telles ned kun i de periodene

mediet er ledig. Det er også definerte en minimum tidsperiode etter hver overføring (dvs. fra mediet blir detektert som ledig) hvor normale data ikke kan sendes, for å sikre at kvitteringer og andre korte kontrollmeldinger får høyere prioritet til å bruke mediet enn data. (2 poeng)

CSMA/CD brukes for klassisk Ethernet der det er mulig å lytte og sende samtidig. (1 poeng)

CSMA/CA brukes for trådløse (radio) nett der det i praksis er vanskelig å detektere kollisjoner; i stedet er en avhengig av eksplisitte kvitteringer (ACK) for å få bekreftet mottak av rammene. (1 poeng)

2.3 Linklagssvitsj / Link layer switch

Gjør rede for hvordan en linklagssvitsj virker. På hvilke måter er den ulik en ruter? (6 p).

Løsningsforslag (med forslag til poenggivning):

Hovedfunksjonen til en linklagssvitsj er å videresende (eller svitsje) innkommende rammer fra et interface ut på ett eller flere andre interface, eller eventuelt filtrere (droppe) rammen hvis den hører hjemme i den retningen den ankom fra. (1 poeng)

Svitsjingen baseres på linklags (MAC) adresser. En svitsjetabell brukes for å avgjøre hvilke utganger (interface) en ramme skal sendes til. Hvis linklagsadressen ikke finnes i tabellen sendes den til alle andre intefaces enn det den ankom via. Hvis adressen finnes men assosieres med samme interface som den ankom via, droppes rammen. Hvis ikke sendes den ellers til det interfacet den assosieres med. (2 poeng)

Linklags svitsjer er selvlærende, ved at det registreres hvilket interface rammer kommer inn via («fra» adressen på linklaget). I tillegg slettes alle tabellinnslag etter en viss tid hvis de ikke brukes, så innholdet er dynamisk. (1 poeng)

I tillegg til det over skiller linklagssvitsjer seg fra rutere ved at de kun opererer internt i et subnett og er transparente for tjenere og rutere, dvs. de har ikke egne adresser slik rutere har (IP adresser på nettverkslaget). Linklagssvitsjer har også den fordelen at de er selvkonfigurerende. (2 poeng)

3.1 Symmetrisk nøkkel og Offentlig nøkkel kryptering / Symmetric key and Public key encryption

Beskriv virkemåte for både det vi kaller symmetrisk nøkkel kryptering ("Symmetric Key Encryption") og det vi kaller offentlig nøkkel kryptering ("Public Key Encryption"). Hva er de viktigste forskjellene? (6 p).

Løsningsforslag (med forslag til poenggivning):

Symmetrisk nøkkel kryptering: De to partene som kommuniserer deler en hemmelig nøkkel slik at informasjon som utvesles kan holdes hemmelig. Denne nøkkelen må formidles mellom dem på en sikker måte, noe som kan være utfordrende. Krypteringsalgoritmen som brukes er vanligvis kjent (i moderne, ikke militære systemer i alle fall) så hele sikkerheten baseres på at nøkkelen er lang nok til at den ikke kan finnes, uansett regnekraft. Samme nøkkel brukes både for å kryptere og for å dekryptere. (2 poeng)

Offentlig nøkkel kryptering: I disse systemene er også algoritmene kjent. Hver part som kommuniserer har nå to nøkler, en hemmelig og en offentlig kjent (og sertifisert; av juridiske grunner er det viktig at den offentlige nøkkelen garanteres å tilhøre den som hevder det; vanligvis bekreftet av en nøytral tredjepart). Hvis to parter nå ønsker å holde kommunikasjonen seg imellom hemmelig, må de bruke hverandres offentlige nøkler til å kryptere informasjon og sende det til den andre, som da er den eneste som kan lese det ved hjelp av sin hemmelig nøkkel. (2 poeng)

Forskjeller: I tillegg til forskjell i virkemåte, som angitt over, har offentlig nøkkel kryptering noen ekstra egenskaper ved at det også kan brukes til å sikre integritet/opphav til meldinger (når det er viktigere enn at noe er hemmelig) og til å lage digitale signaturer, typisk ved å endre rekkefølgen i bruken av nøklene. (Se også nedenfor). (2 poeng)

3.2 Digital signatur / Digital signature

Forklar hva en digital signatur er og hvordan den kan etableres ved bruk av krypteringsmekanismer. (7 p).

Løsningsforslag (med forslag til poenggivning):

En digital signatur er en digital versjon av en vanlig signatur, f.eks. på en avtale eller en kontrakt. Det er svært viktig at en kan stole på en slik signatur, dvs. at den er (tilnærmet) umulig å forfalske. (1 poeng)

Offentlig nøkkel kryptering kan brukes direkte til å signere et dokument digitalt. Hvis noe krypteres ved å bruke en hemmelig nøkkel, kan alle sjekke at dette ble gjort av rette vedkommende ved å bruke den korresponderende offentlige nøkkelen. Integriteten til innholdet i dokumentet sikres da også, dvs. det kan ikke endres etter at signaturen er laget uten at den blir ugyldig. (3 poeng)

Det er viktig at den offentlige nøkkelen beviselig tilhører den som har signert. Dette kan bekreftes av en nøytral tredjepart («Certification Authority» - CA). Dette omtales som «Public Key Certification» og er en viktig del av digital signatur rammeverket. (1 poeng)

I praksis brukes sjelden digitale signaturer på hele (store) dokumenter siden dette kan være prosessorkrevende. Det er derfor mye vanligere at det først lages en

«Message Authentication Code» (MAC) for hele dokumentet, og at det er denne som så signeres i stedet. En slik MAC er en mange-til-en «hash» funksjon (dermed mye kortere enn et helt dokument) som skal være umulig å regne seg tilbake til utgangspunktet fra, dvs. at det er svært vanskelig å endre dokumentet slik at en ender opp med samme MAC. Dette ivaretar derfor fremdeles kravene til integritet av meldingen og at det er digitalt signert. (2 poeng)

3.3 Brannmurer / Firewalls

Gjør rede for de tre typene brannmurer ("Firewalls") som dekkes av pensum, med spesiell fokus på ulikhetene i funksjonalitet. (7 p).

Løsningsforslag (med forslag til poenggivning):

Tradisjonell pakkefiltrering («Traditional packet filters»): Filtrering av alle inn- og utgående informasjonsenheter til/fra en organisasjons ruter(e), basert på definerte filtreringsregler («security/filtering policy»). Vanligvis gjøres filtreringen på basis av adresser og portnummer og kan være ulik for inn- og utgående informasjonsenheter. Tilleggsinformasjon kan også brukes, f.eks. om TCP ACK bit er satt eller ikke i et segment. Alle filtreringsavgjørelser er basert på informasjon i hver enkelt informasjonsenhet. En aksesskontroll liste implementeres og vedlikeholdes basert på disse reglene. (3 poeng)

Tilstandsbasert pakkefiltrering («Stateful packet filters»): I tillegg til funksjonaliteten over holder denne metoden rede på *forbindelser* for å gjøre filtreringsfunksjonen mer effektiv og sikker. En egen liste over active forbindelser benyttes, i tillegg til en (litt utvidet) versjon av den tradisjonelle aksesskontroll listen. (2 poeng)

Applikasjonsgateway («Application gateways»): Her brukes applikasjonsspesifikke tjenere som all innkommende og utgående informasjonsenheter må passere. I tillegg til funksjonaliteten i metodene over kan en nå da også bruke applikasjonsspesifikke regler for å filtrere. En kan også ha adgangskontroll mot enkelte applikasjoner. Logisk kreves en gateway for hver applikasjon i denne modellen, men disse kan dele de samme fysiske maskinene. (2 poeng)

4.1 Lagret versus interaktive media / Stored versus Conversational media

På hvilke måte skiller "Strømming av lagret audio og video" ("Streaming Stored Audio and Video") og "Interaktiv tale og video over IP" ("Conversational Voice- and Video-over-IP") seg fra hverandre med hensyn til de krav som stilles til nettet og ende-bruker utstyret? (5 p).

Løsningsforslag (med forslag til poenggivning):

«Strømming av lagret audio og video»: Dette er i hovedsak transport av data i en retning gjennom nettet (bortsett fra litt kontroll- eller styringsinformasjon i motsatt

retning). Sanntidskravene er små, spesielt hvis mottaker har mulighet for å buffre en del data. Typisk vil en da overføre en del data for å fylle opp et buffer før en starter avspilling hos mottaker for å gjøre videre overføring mindre påvirket av stokastiske variasjoner i forsinkelse i nettet. Kravene til nettet er at en klarer, i middelverdi, å overføre nok data til å gi den kvaliteten som forventes, men tolererer vanligvis litt variasjon i forsinkelse gjennom nettet, gitt at nok bufferplass er tilgjengelig hos mottaker. Litt pakketap er vanligvis akseptabelt, men ikke for mye siden en vanligvis forventer god kvalitet av det som spilles av. Kravene til endebrukerutstyret er at en klarer å spille av i den kvaliteten som kreves (både hos sender og mottaker), pluss at en har nok bufferplass hos mottaker til å utjevne variasjonen i forsinkelse gjennom nettet. (3 poeng)

«Interaktiv tale og video over IP»: I dette tilfellet overføres (generelt) like mye data i begge retninger gjennom nettet. Sanntidskrav gjør også at det er begrenset hva som tåles av buffring før samtalen får kunstig lange pauser. For å unngå forstyrrelser (i lyd og/eller bilde) stilles da også større krav til at forskjellen i forsinkelse gjennom nettet er mindre enn for en-veis strømming. Noe pakketap er likevel vanligvis mer akseptabelt for interaktiv kommunikasjon enn for strømming. Kravene til endeutstyret er (som over) at en klarer å håndtere den kvaliteten som forventes. Bufferplass hos mottaker spiller mindre rolle, siden en ikke ønsker å forsinke avspillingen mye.
(2 poeng)

4.2 Håndtering av pakketap / Handling of packet loss

Gjør rede for de tre metodene (som omtalt i pensum) for å håndtere pakketap i VoIP applikasjoner. (Stikkord: funksjonalitet; hva som oppnås; ressurskrav). (5 p).

Løsningsforslag (med forslag til poenggivning):

Forward Error Control (FEC) med ekstra XOR data: Metoden går ut på å sende en ekstra pakke for hver n-te pakke som overføres. Denne ekstra pakken er konstruert ved å bruke XOR funksjon på bitnivå over alle de n pakkene. Hvis en mister en av de n+1 pakkene som overføres vil en da alltid kunne rekonstruere denne perfekt hos mottaker. En oppnår altså å unngå reduksjon i kvalitet. Siden en ekstra pakke sendes vil kapasitetbehovet øke litt. En vil også måtte bruke litt prosesseringskraft hos mottaker i de tilfellene en pakke må rekonstrueres, noe som også kan medføre en liten forsinkelse (dog litt avhengig av hvor lenge data buffres hos mottaker i utgangspunktet). (2 poeng)

Forward Error Control (FEC) med ekstra lavere rate bitstrøm: Metoden går ut på å sende en lavere kvalitets lydstrøm innimellom de pakkene som inneholder den kvaliteten vi egentlige ønsker. I praksis kan dette f.eks. gjøres ved å utvide størrelsen på hver pakke og legge til et lavere kvalitets lydsampel i hver pakke, med ett sampels forskyvning i forhold til den opprinnelige. Hvis en pakke da mistes vil en bruke lavere kvalitets sampelet i neste pakke i stedet. En oppnår i praksis et godt resultat med denne metoden, selv om kvaliteten på det ene sampelet da er lavere enn det skulle

ha vært. Kapasitetsbehovet øker noe også med denne metoden siden hver pakke blir litt større. Prosesseringsbehovet hos mottaker blir imidlertid minimalt og bør ikke gi noe forsinkelse i avspillingen. (1 poeng)

Interleaving av data: Metoden går ut på å endre sekvensen av talesampler i pakkene som overføres slik at effekten av et pakketap spres utover og får mindre effekt på kvaliteten enn tap av mange påfølgende talesampler ville gi. En oppnår altså redusert kvalitet over litt lengre tid, i stedet for (potensiell) «drop out» i en kortere tidsperiode. Det er inge krav til ekstra kapasitet i nettet ved bruk av denne metoden, men det introduseres forsinkelser både på sender og mottakersiden når sampler må hhv. spres utover eller samles sammen igjen til riktig posisjon i talestrømmen. Metoden er derfor vanligvis dårlig egnet for interaktiv tale. Det kreves også prosessering både hos sender og mottaker. (2 poeng)

NB! Hvis noen studenter har inkludert «Error concealment» som en av de tre metodene i stedet for å dele FEC inn i to sub-typer, bør de få kreditt for det, gitt at de har fått med momentene nedenfor:
«Error concealment»: Enten spille av samme pakke på ny hvis en pakke mistes, siden det vanligvis er stor korrelasjon mellom talesampler; eller Interpolasjon mellom to talesampler (er også er nevnt i pensum). Ingen av disse krever ekstra kapasitet i nettet, men Interpolasjon kan kreve en del prosessering hos mottaker. Begge gir vanligvis akseptabel kvalitet ved lavt pakketap i nettet.
(Alternativt 1 poeng, men maksimum 5 poeng på hele 4.2)

4.3 Forsinkelser i nettet / Delays in a network

Gjør rede for de ulike bidragene til forsinkelse en informasjonsenhet utsettes for når den transporteres gjennom et kommunikasjonsnett. Angi også typiske verdier av bidragene hvis en antar et geografisk begrenset, men delt nett (f.eks. alt innenfor en diameter på 10 km) og høy kapasitet på overføringen ende til ende (f.eks. 100 Mbit/s). Hvilket bidrag vil variere mest som funksjon av tid? (5 p).

Løsningsforslag (med forslag til poenggivning):

De ulike bidragene til forsinkelse gjennom et nett er:

Prosesseringsforsinkelse («Processing delay»): Tiden det tar å prosessere en informasjonsenhet i nettelementer. (0,25 poeng)

Køforsinkelse (Queuing delay»): Tiden en informasjonsenhet må vente på delte ressurser i nettet, eksempelvis adgang til en utgang fra en ruter. (0,25 poeng)

Transmisjonforsinkelse («Transmission delay»): Tiden det tar å sette en informasjonsenhet ut på et fysisk medium. (0,25 poeng)

Propagasjonsforsinkelse («Propagation delay»): Tiden det tar for et bit å bevege seg over et fysisk medium. (0,25 poeng)

Typiske verdier (vi må være åpne for andre måter å estimere på!):

Propagasjonsforsinkelse over en distanse på 10 km er typisk (10 km / 200 000 km/s) = 0,00005 sekund = 0,05 millisekund. (Hvor 200 000 km/s er ca. 2/3 av lyshastigheten).

Transmisjonsforsinkelsen er avhengig av størrelsen på det som overføres (ikke gitt i oppgaven). Hvis vi antar en typisk IP pakke på 1500 bytes vil det ta (1500 x 8) [bit] / 100 000 000 [bit/s] = 0,00012 sekund hver gang pakken legges ut på et medium. Hvis vi antar at det kan skje maksimum 10 ganger i løpet av 10 km, gir det totalt 0,0012 sekund = 1,2 millisekund. Hvis en ønsker å overføre mer enn en typisk IP pakke vil denne selvfølgelig øke tilsvarende.

Prosesseringsforsinkelsen er svært avhengig av hvor rask maskinvare som brukes i nettet. For en moderne ruter maksimum noen mikrosekunder (ifølge læreboken). La oss anta 2 mikrosekunder for en 1500 bytes IP pakke. Med 10 hopp (9 rutere) som over kan vi da anslå et totalt bidrag på 0,018 millisekund.

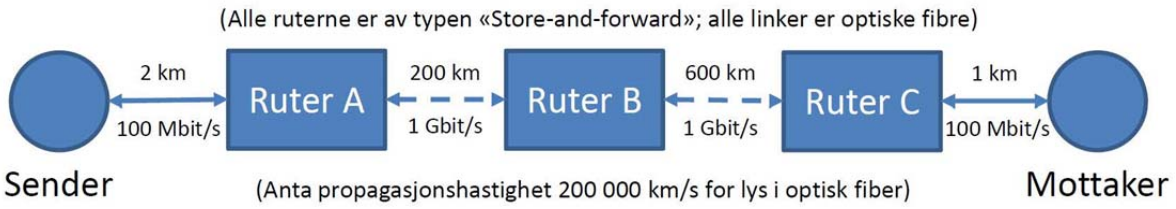
Køforsinkelse er avhengig av hvor mange som deler ressurser til enhver tid og kan derfor variere fra 0 til en svært høy verdi, hvis nettet er overbelastet i perioder.

(Fornuftige resonnementer kan godt avvike fra det over). (2 poeng)

Køforsinkelsen vil være den som variere mest med tiden, siden trafikken i nettet kan endre seg hele tiden, mens de andre forsinkelsene gir nokså konstante bidrag.
(2 poeng)

4.4 Beregninger i et nett / Computations in a network

Gitt en forbindelse som vist på figuren nedenfor:



Anta at du er den eneste som bruker disse linkene, men ruterne kan også brukes av andre forbindelser. Segmentering brukes ikke på denne forbindelsen.

Vis hvordan du beregner følgende:

a) Anta (her i oppgave a)) at propagasjonsforsinkelsen ignoreres. Finn minimum ende-til-ende forsinkelse (dvs. fra Sender til Mottaker) for en pakke på 1200 K bytes. (1 p).
Hva gjør dette til en minimumsverdi, dvs. hvilke bidrag kommer egentlig i tillegg (utover propagasjonsforsinkelsen)? (1 p).

b) Finn propagasjonsforsinkelsen fra Sender til Mottaker. (2 p).

c) Hva blir den totale minimale end-til-ende forsinkelsen? (1 p).

Løsningsforslag (med forslag til poenggivning):

a) Transmisjonsforsinkelsen fra Sender ut på linken til Ruter A er (1200 x 1000 x 8) [bit] / 100 000 000 [bit/s] = 0,096 sekund. Tilsvarende beregning for de neste 3 hoppene gir 0,0096 sekund (fra Ruter A ut på linken til Ruter B, siden kapasiteten er 10 ganger så høy), 0,0096 sekund (fra Ruter B ut på linken til Ruter C), og 0,096 sekund (fra Ruter C ut på linken til Mottaker). Summert får vi altså en minimum ende-til-ende forsinkelse (når propagasjonsforsinkelse og de øvrige ignoreres) på 0,096 + 0,0096 + 0,0096 + 0,096 = 0,2112 sekund. (1 poeng)

   De bidragene som kommer i tillegg er prosessering i ruterne. I dette tilfellet har vi antatt at vi er de eneste som bruker disse linkene, dvs. at (eventuelle) køforsinkelser kun vil være i forbindelse med prosesseringen i ruterne, siden disse er delte ressurser. (1 poeng)

b) Propagasjonsforsinkelsen er gitt direkte av lengden på strekningen som tilbakelegges og propagasjonshastighet for lys i fiber, dvs. (2 + 200 + 600 + 1) [km] / 200 000 [km/s] = 0,004015 sekund. (2 poeng)

c) Den totale minimale ende-til-ende forsinkelsen er da gitt som summen av bidragene over, dvs. 0,2112 + 0,004015 = 0,215215 sekund. (1 poeng)

5.1 Kjerne- versus aksessnett / Core versus access network

Gjør rede for forskjellene mellom et kjernenett og et aksessnett. (5 p).

Løsningsforslag (med forslag til poenggivning):

Aksessnett er nærmest brukerne. Kjernenettet (eller transportnettet) eksisterer for å kunne koble sammen ulike (typer av) aksessnett. Dette gjør det mulig å kommunisere over lengre avstander og mellom ulike typer teknologiske løsninger for kommunikasjon. (2 poeng)

Kjernenettet implementerer kun de tre nederste lagene i protokollstakken, dvs. fysisk lag, linklaget og nettverkslaget. Hvis endesystemer regnes som del av aksessnettet må hele protokollstakken implementeres; men ikke over alt siden et aksessnett også kan inneholde både svitsjer og rutere (som hhv. kun implementerer to eller tre lag av protokollstakken). (2 poeng)

Typiske teknologier som brukes i aksessnett er DSL («Digital Subscriber Line»), Kabel (ofte kombinert med TV), FTTH («Fiber to the home»), Satelitt, Wi-Fi osv. for aksess hjemmefra, eller Ethernet (og WiFi) for aksess fra bedrifter. Typiske teknologier for kjernenett er store pakkesvitsjer (rutere eller linklagssvitsjer) koblet sammen med optisk fiber, eller Linjesvitsjer (f.eks. store optiske crossconnect svitsjer - OXC) og fiber linker. (1 poeng)

5.2 Socket programmering / Socket programming

Forklar hva en oppnår ved å bruke socket programmering ("Socket programming") og gi et eksempel på bruk. (5 p).

Løsningsforslag (med forslag til poenggivning):

Hensikten med socket programmering er at nettbaserte applikasjoner fra potensielt ulike utviklere/bedrifter skal kunne kommunisere over et veldefinert grensesnitt på en standardisert måte. Bruk av portnummer («port numbers») muliggjør også multipleksing på applikasjonslaget, dvs. en PC (f.eks.) kan la mange applikasjoner kommunisere over samme nettforbindelsen «samtidig» (egentlig tidsmultiplekset). (3 poeng)

Det mest åpenbare eksempelet på hvorfor slike standardiserte grensesnitt er viktige er web browsere, som eksisterer i mange ulike varianter men klarer (stort sett) å kommunisere med de samme web serverne. HTTP protokollen, som er sentral for browserne, benytter seg vanligvis av TCP sockets for kommunikasjon. (2 poeng) (Andre eksempler kan også godtas).

5.3 DNS

Gjør rede for hva DNS ("Domain Name System") brukes til i Internet. (Stikkord: virkemåte; funksjonalitet; struktur). (5 p).

Løsningsforslag (med forslag til poenggivning):

Hovedoppgaven for DNS er å tilby en katalogtjeneste som oversetter mellom tjenernavn og IP adresser. (2 poeng)

De to hovedelene DNS består av er en distribuert database implementert som et hierarki av DNS tjenere, og en applikasjonsnivå protokoll som lar tjenerne kommunisere med databasen. (2 poeng)

Det er 13 (logiske) top-nivå («Root») tjenere i DNS, spredd over hele verden for å sikre pålitelighet i operasjonen, siden de er kritiske for at nettet skal virke. Hver av disse er også i realiteten implementert som mange fysiske tjenere, igjen for å øke påliteligheten. Nest øverste nivå av DNS tjenere («Top-level domain») tar seg av hver sine top-domener (f.eks. .org, .com, .edu, og de nasjonale domenene .no, .se, osv.). Neste nivå igjen kalles «autoritære» tjenere («Authoritative DNS servers») og er typisk plassert hos organisasjoner eller store bedrifter for å håndtere egne tjenere, eller for å selge disse tjeneste til andre som ikke selv har denne funksjonaliteten. (1 poeng)

5.4 Nettstruktur / Network structure

Lag en eller flere figurer som illustrerer hvor en linklagssvitsj hører hjemme i et nett. (Se også oppgave 2.3). (Stikkord: protokoll-lag; plassering; adressering). (5 p).

Løsningsforslag (med forslag til poenggivning):
Eksempler på ok figurer i forhold til spørsmålet (tatt fra utgave 6 av læreboken).

Det må vises både protokoll lag (2 poeng) og plassering i nettet / adressering (3 poeng) som illustrert hhv. i de to figurene nedenfor.

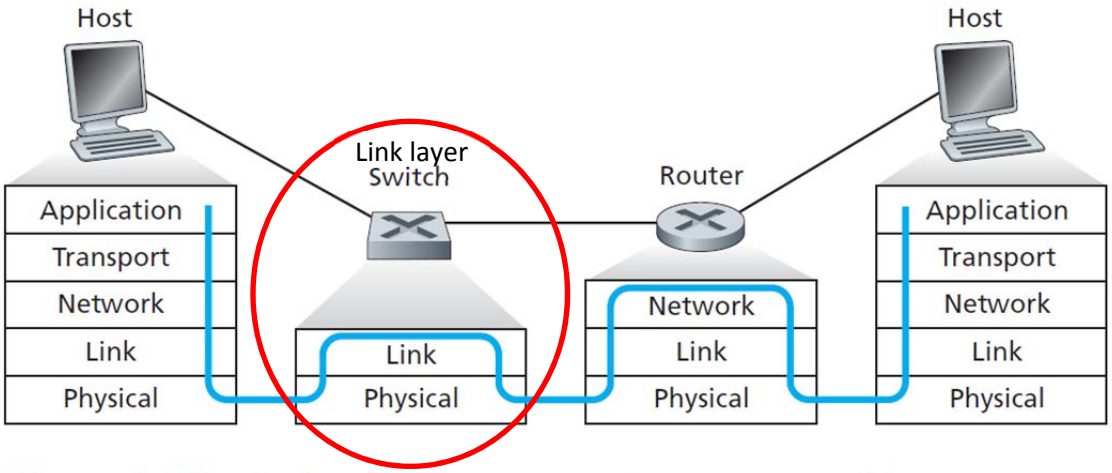


Figure 5.24 ♦ Packet processing in switches, routers, and hosts

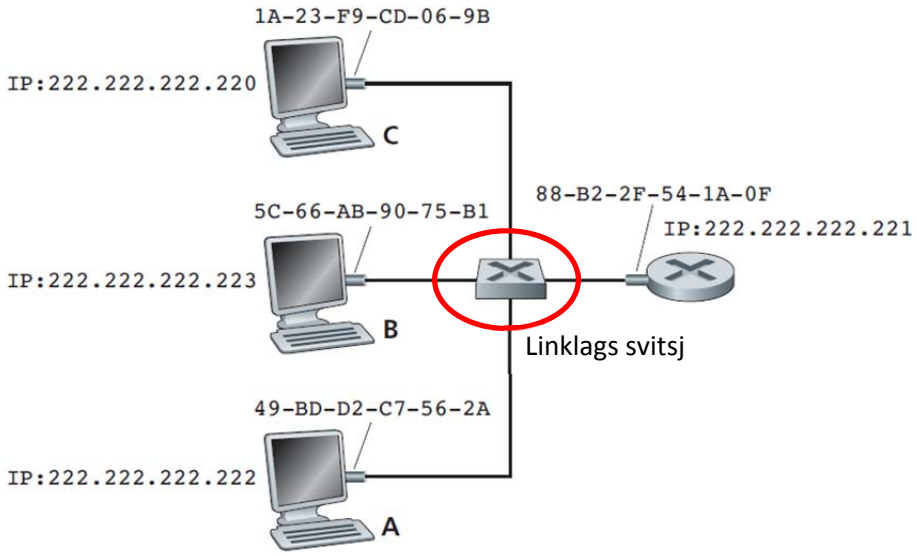


Figure 5.17 ♦ Each interface on a LAN has an IP address and a MAC address