Løsningsforslag eksamen TTM4100 14. mai 2021 – versjon 15-5-2021

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 vi mangler i løsningsforslaget.

1.1 Protocol layers (6 p)

Gi en oversikt over protokoll-lags modellen som brukes for kommunikasjon over Internett og forklar spesielt hva som menes med begrepet "innkapsling" ("encapsulation") i denne sammenhengen.

Løsningsforslag (med forslag til poenggivning):

Definert for å gi struktur til design av nettverksprotokoller. Hver protokoll sies å tilhøre et gitt lag i et protokollhierarki og benytter tjenester fra lagene under i protokollhierarkiet. Fra top til bunn er lagene som følger (navn på dataenhet i parentes): Applikasjon (melding («message»)), Transport (segment), Nett (datagram), Link (ramme («frame»)), og Fysisk (overføring av bit). (3p)

Innkapsling: Et nytt protokollhode blir lagt til en dataenhet når den beveger seg nedover i protokollhierarkiet, eller fjernet når den beveger seg oppover. Et eksempel: En ramme på linklaget vil består av et linklagshode og et datagram (fra nettlaget over) som nyttelast («payload»). Et datagram består av et nettlagshode og et segment (fra transportlaget over) som nyttelast, osv. (3p)

1.2 World Wide Web (7 p)

Forklar med egne ord (på høyt nivå) kommunikasjon over World Wide Web (WWW). (Stikkord: Protokoll brukt og egenskapene til denne; caching; cookies).

Løsningsforslag (med forslag til poenggivning):

«HyperText Transfer Protocol» (HTTP) er applikasjonslags-protokollen som brukes for World Wide Web (WWW). Den er implementert i to ulike applikasjonsprogrammer: en klient applikasjon (vanligvis en «Web browser») og en tjener applikasjon («Web server»). Disse to applikasjonsprogrammene kommuniserer med hverandre via HTTP. (2p)

Protokollen er tilstandsløs, noe som betyr at applikasjonen på tjenersiden («Web serveren») ikke vedlikeholder noe informasjon om spesifikke klienter. (1p)

HTTP bruker TCP som transportlagsprotokoll. (1p)

Både persistente («persistant») og ikke-persistente («non-persistant») forbindelser støttes av HTTP. Persistent betyr at et client- og et tjenerprogram (hhv. “Web browser” og “Web server”) kommuniserer (via en oppsatt TCP forbindelse) over en tidsperiode, mens ikke-persistent betyr at en ny TCP forbindelse må settes opp for hver ny «request/response» interaksjon mellom klient og tjener. (1p)

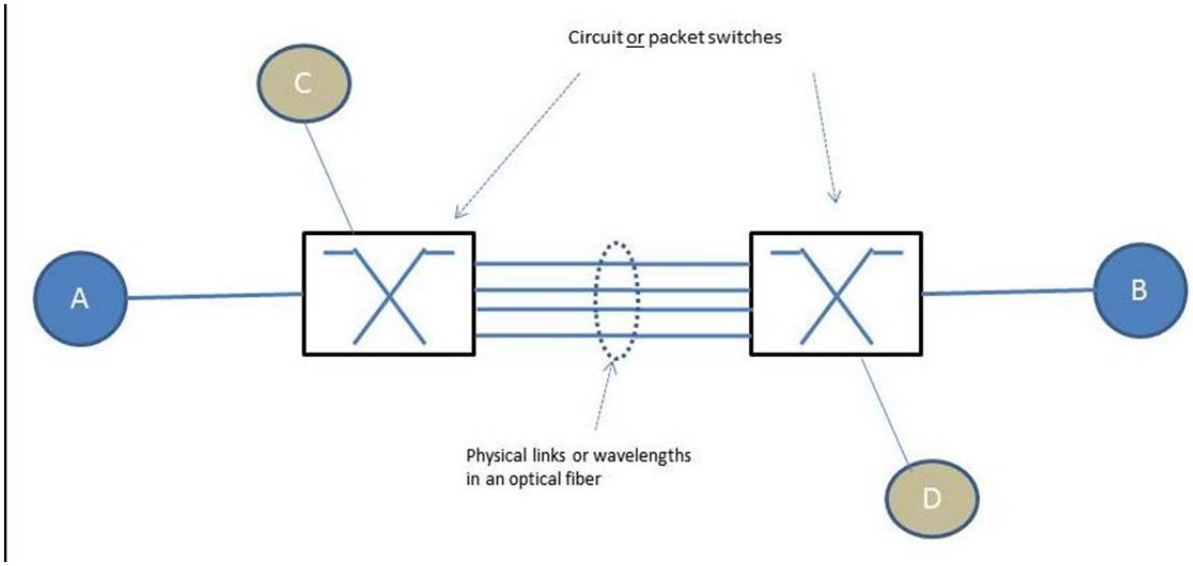
«Cookies» er informasjon som lagres delvis lokalt hos en klient og i serverens database for å motvirke at HTTP er tilstandsløs. Tjenerapplikasjonen («Web server»)

kan da i framtidig kommunikasjon bli meddelt en pekerverdi (i form av et stort unikt heltall som identifiserer denne kunden) fra klienten («Web browseren») og benytte seg av denne informasjonen til å finne riktig informasjon i egen database, for å slippe «å starte på ny» hver gang de kommuniserer. (1p)

«Web caching» brukes i nettet for å redusere trafikk og prosessering i WWW. Disse er til stede i de fleste klient applikasjoner («Web browsere»), for å slippe å laste ned sider på ny dersom de ikke har endret seg siden sist de ble lastet ned; dvs. en lagrer en lokal kopi av siden som kan brukes i stedet. I tillegg er det spesialiserte «Web cacher» også i nettet (f.eks. Content Distribution Networks – CDN»), hvor de mest populære websidene og innholdet lagres. I tillegg til å redusere trafikk og prosessering, minsker det også ventetider for å aksessere innhold over nettet. (1p)

1.3 **Delay in networks** (7 p)

Se figuren nedenfor.



Anta at kommunikasjon mellom A og B i nettet kan være enten pakkesvitsjet (basert på "store-and-forward" prinsippet) eller linjesvitsjet. Anta at det allerede er satt opp en forbindelse mellom A og B. Det er også andre aktive forbindelser i nettet, f.eks. mellom C og D vist i figuren.

Gjør rede for de ulike bidragene til forsinkelse gjennom nettet når en bruker

a) linjesvitsjing for en informasjonsenhet som sendes fra A til B.

b) pakkesvitsjing for en informasjonsenhet som sendes fra A til B.

Løsningsforslag (med forslag til poenggivning):

De ulike bidragene til forsinkelse gjennom et nett kan være: (1p - hvis bra oversikt over *mulige* bidrag)

Prosesseringsforsinkelse («Processing delay»): Tiden det tar å prosessere en informasjonsenhet i nettelementer. (Kan også inneholde en ørliten køforsinkelse siden prosessoren(e) også er delt(e) ressurs(er)).

Køforsinkelse (Queuing delay»): Tiden en informasjonsenhet må vente på delte ressurser i nettet, eksempelvis adgang til en utgang fra en ruter. Kan være høy ved stor trafikk.

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

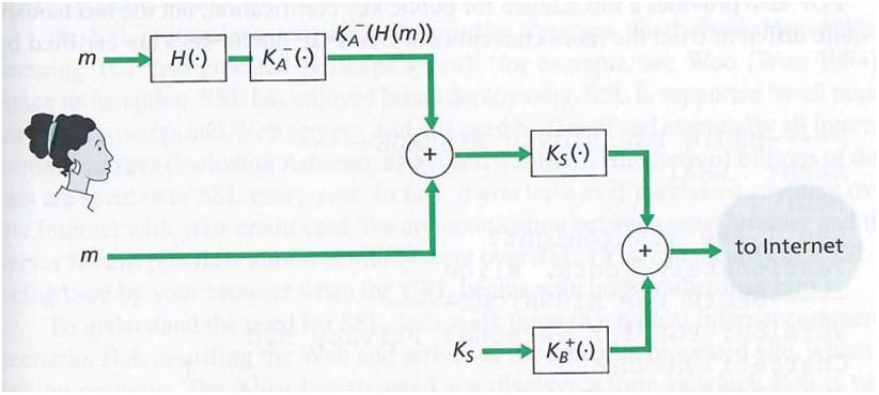
Propagasjonsforsinkelse («Propagation delay»): Tiden det tar for et bit å bevege seg over et fysisk medium.

a) Når linjesvitsjing benyttes for en informasjonsenhet på en allerede oppsatt forbindelse, vil denne ha en reservert fysisk kanal fra A til B. Dvs. den konkurrerer ikke med noen andre informasjonsenheter i nettet, og transmitteres tvers igjennom de to svitsjene i figuren. De eneste bidragene til forsinkelse i dette tilfellet er: (3p samlet for hele a))
    a. En transmisjonforsinkelse når informasjonsenheten settes ut på linjen av A.
    b. Propagasjonsforsinkelse fra A til B, på reserverte gjennomkoblinger gjennom de to linjesvitsjene.
b) Når «store-and-forward» pakkesvitsjing brukes og det er konkurrerende trafikk om de samme ressursene, får en bidrag fra alle typene forsinkelser over: (3p samlet for hele b))
    a. Tre transmisjonsforsinkelser; en når informasjonsenheten settes ut på linja av A, samt to til når den settes ut på linjen av hver av de to pakkesvitsjene. «Store-and-forward» betyr at hele pakken må mottas av pakkesvitsjen før den kan prosesseres (f.eks. feilsjekk; adressesjekk for å finne riktig utgang fra pakkesvitsjen).
    b. Propagasjonsforsinkelsen fra A til B blir omtrent det samme som for linjesvitsjing, hvis vi antar at signalveiene internt i pakkesvitsjene er korte sammenlignet med de andre avstandene.
    c. To prosesseringsforsinkelser; en i hver av pakkesvitsjene.
    d. To køforsinkelser; en ved utgangen fra hver av pakkesvitsjene siden disse er delte ressurser.

(Eventuell prosessering i A og B hhv. før sending og etter mottak av informasjonsenheten antas lik i de to tilfellene – regnes vanligvis å være utenfor nettet).

1.4 **E-mail security** (7 p)

Se figuren nedenfor som illustrerer (deler av) en prinsipiell implementasjon av sikker e-post utveksling.



Forklar med egne ord hvilke operasjoner som utføres i figuren og hva som oppnås med disse. Forklar også hvilke operasjoner som må legges til for å realisere sikker e-post utveksling ende-til-ende.

Løsningsforslag (med forslag til poenggivning):

På høyt nivå: operasjonene øverst til venstre i figuren bidrar til autentisering av avsender og integritet av meldingsinnholdet, mens operasjonene til høyre bidrar til konfidensialitet av meldingsinnholdet. (2p)

Mer detaljert: I øverste linje lages en hash funksjon H(.) av meldingen m, for deretter å signere denne hash funksjonen med avsenders private nøkkel $K_A^-(.)$ for å lage en digital signatur. Resultatet er $K_A^-(H(m))$, som kombineres (serielt) med meldingen selv (m) i «+» funksjonen til venstre. På dette tidspunktet kan både sender autentiseres og meldingens integritet sjekkes, men selve meldingen er fremdeles i klartekst. (1p)

For å også oppnå konfidensialitet av meldingen (og den digitale signaturen) må disse også krypteres med en tilfeldig symmetrisk sesjonsnøkkel $K_S(.)$. Selve sesjonsnøkkelen må også overføres kryptert til mottaker. Det gjøres ved å kryptere den med mottakers offentlige nøkkel $K_B^+(.)$. Dette kombineres (serielt) med resultatet over i «+» funksjonen til høyre, før alt sendes til mottaker via internet. (1p)

Hos mottaker (ikke vist i figuren) må de «motsatte» operasjoner gjøres:

1) Bruke sin private nøkkel for å finne sesjonsnøkkelen (valg av sender).
2) Bruke sesjonsnøkkelen til å dekryptere meldingen og den digitale signaturen.
3) Bruke senders offentlige nøkkel til å sjekke at hashfunksjonen mottatt stemmer overens med den hashfunksjonen han selv får for meldingen. Hvis ja er både meldingintegritet og avsender verifisert. (2p)

Disse operasjonene er likevel avhengige av at de offentlige nøklene til både avsender og mottaker er korrekte. Dette kan f.eks. sikres ved å benytte en «Certificate Authority - CA» som begge stoler på. (1p)

2.1 General functionality (6 p)

Gi en oversikt over transportlaget for kommunikasjon over Internet. (Stikkord: hovedoppgaver/funksjoner, protokoll(er) brukt, hvor i nettet det er til stede).

Løsningsforslag (med forslag til poenggivning):

Transportlagets hovedoppgave er å tilby en (logisk/virtuell) ende-til-ende forbindelse til applikasjoner som befinner seg på ulike steder i nettet. Sett fra applikasjonene kommuniserer de direkte via en transportlagsprotokoll, dvs. lavere lag i protokollhierakiet er usynlige. (2p)

«Transmission Control Protocol» (TCP) og «User Datagram Protocol» (UDP) er de to vanligste protokollene på transportlaget. UDP tilbyr forbindelsesløs kommunikasjon, mens TCP tilbyr forbindelsesorientert kommunikasjon hvor det settes opp en virtuell forbindelse mellom de to endesystemene som vil kommunisere, og har mekanismer for flytkontroll og overlastkontroll. (2p)

Siden transporlaget oppgave er å kommunisere mellom to endesystemer er laget kun tilgjengelig i endesystemene, ikke inne i nettet. (2p)

2.2 Sequence numbers and ACK (7 p)

Anta en etablert TCP forbindelse mellom vertene A og B. På et gitt tidspunkt i kommunikasjonen har A mottatt fra B alle data opp til og med byte 433. Anta at Vert B deretter sender to segmenter med data til Vert A uten opphold mellom dem («back-to-back»). Første og andre segment inneholder henholdsvis 15 og 50 bytes med data. I det første segmentet er sekvensnummeret 434, kildeportnummeret er 495, og destinasjonsportnummeret er 344. Vert A sender alltid en kvittering når den mottar et segment fra Vert B.

a) Hva er sekvensnummeret, kildeportnummeret og destinasjonsportnummeret i det andre segmentet som sendes fra Vert B til Vert A?

b) For samme situasjon som beskrevet over: Hvis det andre segmentet sendt ankommer til vert A før det første segmentet, hva er kvitteringsnummeret i kvitteringen (fra A til B) for det segmentet som mottas først?

Løsningsforslag (med forslag til poenggivning):

a) Sekvensnummeret er 434 + 15 = 449, dvs første byte i neste segment. Portnumrene endres ikke siden det er kommunikasjon i samme retning, dvs. kildeportnummer er 495 og destinasjonsportnummer er 344. (3p)
b) Siden en mottaker alltid gir beskjed om første utestående oktett den venter på, vil kvitteringsnummeret være 434, siden denne byten ikke er mottatt ennå. (4p)

2.3 **TCP congestion control** (7 p)

Gi en oversikt over hvordan overbelastningkontroll ("congestion control") er implementert i TCP protokollen. (Stikkord: tre hovedmekanismer; hovedformål og funksjonalitet for hver av disse).

Løsningsforslag (med forslag til poenggivning):

For TCP håndteres overbelastningskontrollen 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). (4p)

De tre hovedmekanismene brukt er «Slow start», «Congestion avoidance» og «Fast recovery». Algoritmene som styres disse er relativt kompliserte og mange hjelpeverdier benyttes i vurderingene («congestion window» - cwnd, «slow start threshold» - ssthresh). Detaljer i hvordan hjelpeverdiene brukes er utelatt nedenfor.

Formålet og funksjonalitet for hver:

«Slow start»: Som navnet sier prøver TCP å gradvis øke raten segmenter sendes inn i nettet med. Likevel vil raten etter hvert økes eksponensielt hvis nok kvitteringer mottas, så navnet er kanskje litt misvisende. Mekanismen restarter hvis timeout observeres. Hvis en når et nivå lik halvparten av når den observerte overbelastning sist gang, avsluttes «slow start» og «congestion avoidance» starter i stedet. (1p)

«Congestion avoidance»: I denne fasen økes raten segmenter sendes inn i nettet med mye mer forsiktig (lineært) enn i «slow start» for å prøve å unngå overbelastning. Ved indikasjon på overbelastning (f.eks. «triple duplicate ACK») justeres hjelpevariablene (til litt ulike verdier avhengig av TCP versjon) og «fast recovery» startes (hvis den brukes – den er opsjonell, men de nyeste versjonene av TCP bruker den). (1p)

«Fast recovery»: Poenget her er å prøve å unngå å gå helt tilbake til «slow start» hvis overbelastningen er lav. Det er mulig å gå både tilbake til «congestion avoidance» og til «slow start» fra denne fasen, avhengig av hvor sterke indikasjonene på overbelastninglast er. (1p)

Til sammen utgjør disse tre fasene noe som kalles «additive-increase, multiplicative-decrease» (AIMD) form for overbelastningskontroll, med en typisk sagtann oppførsel.

2.4 **SSL** (7 p)

Gi en oversikt over hensikten med og implementasjonen av Secure Socket Layer (SSL). (Stikkord: tre faser; hva oppnås i hver fase; "nonces"; MAC).

Løsningsforslag (med forslag til poenggivning):

SSL er laget for å legge til sikkerhetsfunksjoner på transportlaget for TCP forbindelser. (2p)

I protokollhierarkiet er SSL plassert på grensen mellom applikasjons- og transportlaget siden en applikasjon som skal benytte SSL bruker «SSL sockets» i stedet for «TCP sockets» (Men SSL sublaget bruker «TCP sockets»). (1p)

De tre fasene som brukes i SSL er «Handshake», «Key derivation» og «Data transfer». (1p)

«Handshake»: det som oppnås i denne fasen er å autentisere hvem en kommuniserer med, samt å utveksle en master-nøkkel («master secret»), som senere brukes til å lage flere symmetriske nøkler for kommunikasjon. Offentlige nøkler fra Certification Authority (CA) er viktig å bruke her. Nonces må også brukes, for å hindre «man-in-the-middle» angrep (her: «connection replay» angrep). (1p)

«Key derivation»: master-nøkkelen brukes til å generere 4 ulike symmetriske nøkler som brukes for kommunikasjon, to for å kryptere data og to brukt for å verifisere data (MAC). I tillegg utveksles MAC tatt over alle meldingene utvekslet under «handshake», for å sikre at informasjon ikke har blitt slettet av en «man-in-the-middle». (1p)

«Data transfer»: For å kunne verfisere integriteten til data underveis i kommunikasjonen deles bytestream dataene i TCP opp i mindre biter («records») som hver verifiseres med en MAC. På den måten sjekkes det hele tiden underveis i kommunikasjonen at data ikke er endret på. For å unngå «man-in-the-middle» angrep må en også bruke sekvensnummer når MAC beregnes (for å hindre gjentak av en gyldig «record»). (1p)

3.1 **IPv4 fragmenting** (7 p)

a) Hva menes med fragmentering (i Internet protokoll sammenheng) og hvorfor brukes det for IPv4 datagrammer?

b) Hvor blir fragmenter reassemblert (“reassembled”) når IPv4 brukes?

Løsningsforslag (med forslag til poenggivning):

a) Fragmentering er å dele opp et IP datagram i to eller flere mindre IP datagram, innkapsle hver av disse mindre delene i en linklags ramme, og sende disse rammene til utgående link. Dette er nødvendig fordi ulike linklagsprotokoller har ulike maksimale størrelser på rammene de kan frakte, f.eks. gitt av ulike fysiske begrensninger på ulike fysiske media. (4p)
b) For IPv4 kan fragmenteringen gjøres av rutere, men reassemblering («reassembly») gjøres kun i endesystemene. Fragmentene merkes ved å bruke et «Flag» bit i IPv4 headeren. Flagget har verdi 1 for alle bortsett bortsett fra det siste fragmentet som har 0, slik at endesystemet vet når den kan reassemblere. (3p)

3.2 IPv4 versus IPv6 (6 p)

Gjør rede for de viktigste forskjellene mellom IP versjon 4 og IP versjon 6.

Løsningsforslag (med forslag til poenggivning):

Hovedgrunnen til at en ny versjon av IP-protokollen ble utviklet, var erkjennelsen av at adresseområdet for IP versjon 4 på et tidspunkt ville bli for lite. Men som vist nedenfor, har en benyttet sjansen til å også implementere noen andre endringer/forbedringer i IP versjon 6:

- Adresseområde: Økt fra 32 bit adresser til 128 bits adresser. (2p)
- Prosessering i rutere: siden rutere (i det minste for øyeblikket) er elektroniske, er det viktig å optimalisere behandlingen av pakker for både skalerbarhet og energibruk. Noen bidrag til dette:
  - Et mer strømlinjeformet 40 byte IP-pakkehode er lettere å behandle. (1p)
  - Ikke la rutere gjøre fragmentering, men overlate dette til kun endesystemene. (1p)
  - Fjerning av sjekksumen for IPv6 pakkehode. Det antas at dette blir tilstrekkelig ivaretatt av transport- og linklagene. (1p)
- Merking av dataflyt («Flow labelling»): Det er ikke helt klart hva dette skal brukes til, men det kan være viktig i fremtidig kommunikasjon, f.eks. for å gi forskjellige tjenestekvalitet til forskjellige typer trafikk i nettet. Merking av dataflyter lar deg holde oversikt over tilkoblinger også i nettverkslaget, dvs. i rutere, ikke bare ende-til-ende på transportlaget, som med TCP. (1p)

3.3 IPv4 addressing (7 p)

Anta at en ruter i nettet har følgende CIDR ("classless inter-domain routing") innslag i rutingstabellen:

Address/mask	Next hop
135.46.0.0/22	Interface 0
135.46.128.0/22	Interface 1
192.53.40.0/23	Interface 2
Default	Interface 3

I hvilken retning ut av svitsjen ("Next hop") sendes følgende ankommende IP pakker?

a) 135.46.129.10
b) 135.46.0.14
c) 135.46.48.2
d) 192.53.40.7
e) 192.53.56.7

Løsningsforslag (med forslag til poenggivning):

En måte å løse dette på er å først skrive de aktuelle delene av adressene (dvs. tredje digitale skrives ut binært, de andre beholdes som digitale) som skal «matches» mot i tabellen digitalt, dvs:

135.46.0.0/22 = 135.46.000000|00.0 (vertikal strek angir grense for subnettet).

135.46.128.0/22 = 135.46.100000|00.0

192.53.40.0/23 = 192.53.0010100|0.0

(«Default» er alle adresser som ikke matcher noen av de over)

Vi gjør tilsvarende med adressene som skal matches:

a) 135.46.129.10 = 135.46.10000001.10
   Vi ser at denne matcher 2. linje i rutingstabellen, dvs. den sendes til **Interface 1**. (2p)
b) 135.46.0.14 = 135.46.00000000.14
   Vi ser at denne matcher 1. linje i rutingstabellen, dvs. den sendes til **Interface 0.** (1p)
c) 135.46.48.2 = 135.46.00110000.2
   Vi ser at denne ikke matcher noen av de tre første tabellinnslagene. Den blir da «Default» som sendes ut på **Interface 3**. (2p)
d) 192.53.40.7 = 192.53.00101000.7
   Vi ser at denne matcher 3. linje, dvs. den sendes til **Interface 2**. (1p)
e) 192.53.56.7 = 192.53.00111000.7
   Vi ser at denne ikke matcher noen av de tre første tabellinnslagene. Den blir da «Default» som sendes ut på **Interface 3**. (1p)

4.1 General functionality (6 p)

Gi en oversikt over linklaget for kommunikasjon over Internet. (Stikkord: hovedoppgaver/funksjoner, protokoll(er) brukt, hvor i nettet det er til stede).

Løsningsforslag (med forslag til poenggivning):

Linklagets hovedoppgave er å transportere rammer mellom enheter i nettet, dvs. over en link om gangen. (3p)

Ulike typer fysiske linker trenger ulike protokoller. Av den grunn er det veldig mange linklagsprotokoller i bruk, f.eks. Ethernet (med eller uten CSMA/CD), CSMA/CA (for 802.11 trådløs kommunikasjon), PPP («point-to-point»), og mange protokoller for deling av medier på ulike måter som TDM, FDM, CDMA, osv. (2p)

Linklaget er til stede i alle nettelementer. (1p)

4.2 2-dim parity versus CRC (7 p)

To av metodene brukt for feildeteksjon og (i noen grad) korreksjon er 2-dimensjonal paritetssjekk og Cyclic Redundancy Check (CRC). Gjør rede for hva som kan oppnås med disse to metodene. Hva er hovedforskjellen(e) mellom dem?

Løsningsforslag (med forslag til poenggivning):

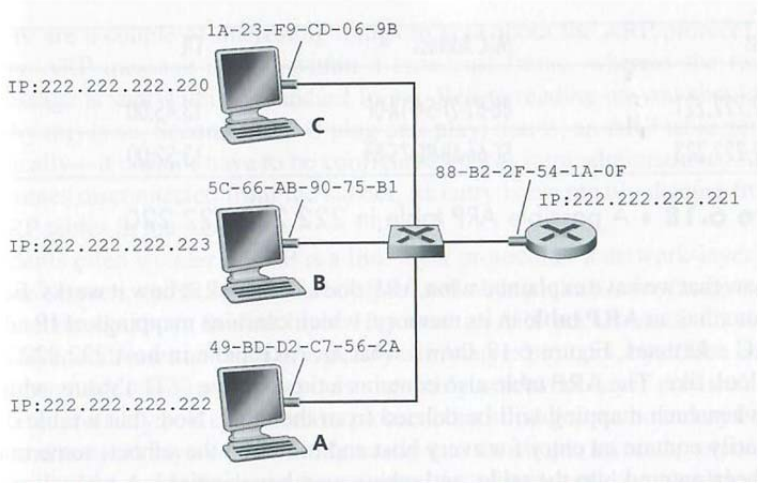
2-dimensjonal paritetssjekk: I dette tilfelle taes paritet i to retninger for hvert bit i kommunikasjonen, noe som betyr at en kan detekterer posisjonen til et bit som er feil, ikke bare det at det er en bitfeil i en gruppe bit eller et ord, som ved 1-dimensjonal paritet. Dette gjør da også at enkeltbitfeil kan korrigeres. Dette er en enkel versjon av «Forward Error Correction» (FEC). Hvis det er to bitfeil vil det også kunne detekteres, men ikke korrigeres. Flere bitfeil vil også kunne detekteres, men ikke alle kombinasjoner av feil. (2p)

Cyclic Redundancy Check (CRC): Denne metoden er standardisert i mange ulike versjoner, med ulikt antall bit i «generatorene» som brukes. Metoden er vellegnet til å detektere «burst» feil, dvs. mange bitfeil på kort tid, og kan detektere like mange av disse som lengden på generatoren minus 1 (hvis en holder seg til standardene). Metoden er også vellegnet for implementasjon i hardware (som shiftregistre) på linklaget. (2p)

Hovedforskjellen på de to metodene er at 2-dimensjonal paritet faktisk kan rette opp enkeltbitfeil (FEC), noe CRC ikke kan. På den annen side er nok CRC bedre egnet til å detektere mange feil på kort tid (burstfeil) enn det 2-dimensjonal paritet er. (3p)

4.3 **ARP** (7 p)

Se figuren nedenfor.



a) Hvorfor trenger vi linklagsadresser (MAC adresser) i tillegg når vi allerede har IP-adresser (på nettverkslaget)?

b) Hvorfor har ikke linklagssvitsjen i figuren noen adresser?

c) Hva er ARP og hvorfor er den nødvendig?

Løsningsforslag (med forslag til poenggivning):

Svar a): En linklagsadresse (MAC adresse) er globalt unik for et nettverksadapter, i motsetning til en IP adresse som avhenger av geografisk plassering. (Personnummer versus gateadresse for en person brukes som analogi i læreboken). Linklagsadressen er den faktiske (fysiske) adressen som er nødvendig å bruke lokalt på et subnet for å komme fram til riktig nettverksadapter. (3p)

Svar b): Den er transparent/usynlig for alle verter og rutere som er tilkoblet den og adresseres aldri av noen. Dens oppgave er å videresender alle rammer den mottar i en eller alle retninger, avhengig av om den finner mottakeren i sin lokale «forwarding» tabell eller ikke. Alternativt kaste en ramme hvis tabellen sier at den skal sendes tilbake i samme retningen som den kom fra. Den er også selvkonfigurerende. (2p)

Svar c): ARP er «Address Resolution Protocol». Den brukes for å oversette mellom IP- og linklags-adresser (i praksis ofte Ethernett-adresser) lokalt i et subnett. ARP tabellene er plassert i minne til hver vert og ruter. (2p)

4.4 W-LAN (6 p)

a) MAC protokollen CSMA/CA for trådløse nett basert på standarden 802.11 har definert ulike varianter av "Inter-Frame Spacing" (f.eks. "Short IFS - SIFS" og "Distributed IFS - DIFS") med ulik varighet. Hva oppnås i CSMA/CA ved å la SIFS være kortere enn DIFS?

b) "Request-to-Send (RTS)" og "Clear-to-Send (CTS)" er et opsjonelt tillegg til 802.11 MAC. Hva kan oppnås med denne mekanismen, og i hvilke tilfeller bør den benyttes?

Løsningsforslag (med forslag til poenggivning):

a) De ulike IFS-ene angir hvor lenge en må vente i tillegg etter at mediet egentlig antas ledig. Ved å bruke «Short IFS» for kvitteringer (ACK) i nettet gies disse prioritet til å bli overført før vanlige datapakker. Siden alle rammer må kvitteres er denne prioriteringen viktig. (SIFS brukes også for de korte kontrollpakkene RTS og CTS, om de brukes). (3p)

b) To trådløse stasjoner som begge kan kommunisere med aksesspunktet (AP), kan ikke nødvendigvis “høre” hverandre, hvis de eksempelvis er på diamentralt motsatte sider i forhold til AP. I slike tilfeller kan en stasjon ikke høre at den andre sender og vil starte å sende selv også, noe som medfører kollisjoner nær AP. RTS og CTS rammer kan da brukes til å reservere kanalen på forhånd. CTS rammen («kvittering» på RTS) vil detekteres av alle stasjoner som kan kommunisere med AP, siden den sendes ut av AP. (2p)

   Selv om denne mekanismen motvirker kollisjoner fra “hidden terminal” situasjonen, introduseres også forsinkelse siden kanalen må reserveres først, og RTS/CTS kontrollrammene bruker også kanalressurser. Av denne grunn brukes mekanismen kun når lange datarammer skal sendes. Hver stasjon kan definere en grenseverdi som sier at en ramme må være over en viss størrelse før RTS/CTS kan brukes. (1p)