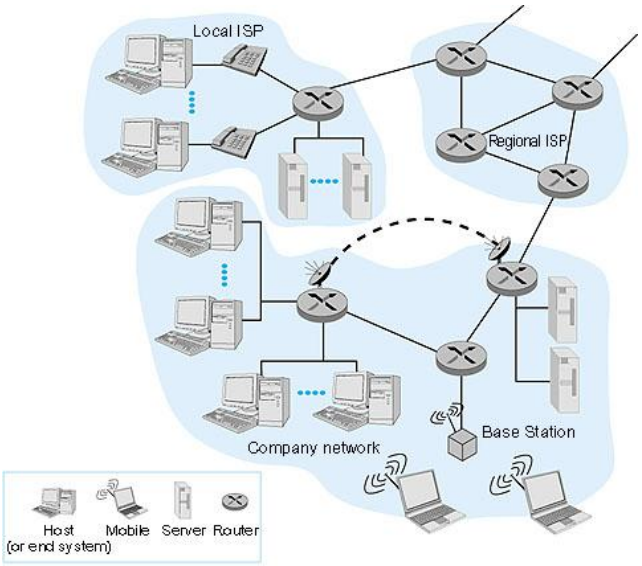
UNIVERSITETET I OSLO

Pensumoppgaver Datakommunikasjon

INF3190
DATAKOMMUNIKASJON

Laget av Khiem-Kim Ho Xuan



Computer networks and the Internet

Referanse: Kap 1

Prinsipper og begreper

Oppgave 1:

Hva er en protokoll?

En protokoll i datamaskinsammenheng er et konvensjonelt(vanlig) eller standardisert sett med regler som bestemmer tilkobling, kommunikasjon og dataoverføring mellom to endepunkter (f.eks. to dataprogrammer på ulike maskiner i et nettverk). Protokoller kan implementeres i maskinvare, programvare, eller en kombinasjon av disse. På det laveste nivået definerer en protokoll en maskinvaretilkobling.

Eks:

IP – Internet Protocol

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

FTP – File Transfer Protocol

HTTP – Hypertext Transfer Protocol

De fleste kommunikasjonsprotokoller er fastsatt av RFC-dokumenter publisert av Internet Engineering Task Force (IETF).

Oppgave 2:

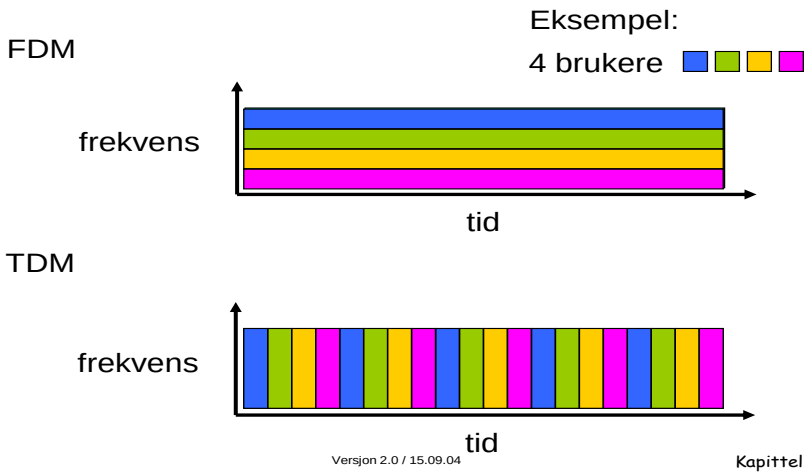
Hva er forskjellen på pakkesvitsjing og kretssvitsjing? Gi eksempler.

Det er to fundamentale tilnærminger når en skal bygge ei nettverkskjerne:

Kretssvitsjing:

- *Mest brukt i forbindelse med vanlig telefoni(ISDN), men kan også brukes til å overføre data*
- *Det etableres en forbindelse(en krets) før overføring finner sted og avsluttes når overføringen er ferdig*
- *Sender og mottaker kobles sammen med en dedikert linje*
- *Reservering av ressurser ende til ende for en forbindelse*
- *Ressursbit ligger ubrukt hvis den ikke brukes av forbindelsen som eier den (ingen deling med andre) feks i pause i talestrøm*
- *Fordel: Linja er "oppe" hele tiden, fast overføringsrate*
- *Ulempe: dårlig utnyttelse av båndbredde*
- *To typer:*
    - ***FDM** (frekvency division multiplexing) fast båndbredde – hver bruker blir tildelt et bestemt frekvensområde*

- ***TDM*** *(time division multiplexing) fast båndbredde - blir delt inn timesloter. Bluetooth*



Pakkesvitsjing:

- *Hver ende- til ende datastrøm deles opp i pakker*
- *Hver pakke bruker hele båndbredden til linken*
- *Ressurser benyttes etter behov*
- *Summen av ressursbehovene kan overskride det som er tilgjengelig; pakker må ligge i kø(buffer) og vente på å bli sendt*
- *Fordeler: tillater flere brukere av gangen enn kretssvitsjing, trenger ikke samtaleoppsett*
- *Ulemper:Kan få metning i buffer -> pakketap(men kan bruke protokoller som håndterer dette, for eksempel TCP)*
- *To typer:*
    - ***Datagram*** *– viss det blir brudd i ene veien, så vil pakkene ta en annen vei. Ruteren finner ut hvor pakken skal. Typisk IP. Fordel at det gir en fleksibilitet, rendundans. Men vi bruker litt lenger tid, IP-adressen er mange flere byte enn ID'ene. Alle kommer i riktig rekkefølge. For eksempel Ethernet.*
    - ***Virtuell krets*** *– En virtuell kobling mellom en sender og mottaker. Har fremdeles pakker (og deler opp pakkene), men kjører en call-setup. Det settes opp en forbindelse, og alle pakkene går samme vei(I motsetning til datagram der pakkene kan ta forskjellige ruter). Hver pakke inneholder en merkelapp ("tag", virtual circuit ID), tag bestemmer neste hopp Typisk er ATM asynkron transfer mode(fast pakkestørrelse). Kan gies mer båndbreddegarantier. Det gjør ikke datagramnettverkene. Ofte backbone (ligger som ryggradsnettverk)*

Oppgave 3:

Hvordan klarer en tjenesteleverandør (ISP) å levere internett aksess til kundene sine (stikkord: NIX, backbone, peering-avtaler, ruting)?

En internettleverandør (forkortes ISP fra engelsk Internet service provider) er et selskap som tilbyr personer eller selskaper tilgang til (leie av) Internett- og andre relaterte tjenester som f.eks. oppbygging av websider og e-postkontoer på sin server.

Tjenestene leveres fra NIX og helt frem til kunden. NIX (Norwegian Internet eXchange) er et norsk knutepunkt for IP-samtrafikk. På NIX kan internettleverandører (ISP-er) koble seg opp, enten ved utplassering av routere på NIX-lokasjonen (kun tilgjengelig på NIX1 og NIX2), eller ved fibertilknytning til eget datasenter, og oppnå samtrafikk/peering med andre internettleverandører. NIX er derfor ikke all internett-trafikk i Norge, men kun trafikk som utveksles mellom forskjellige ISP-er.

En ISP kan enten ha egen NIX-tilknytning og eget IP-nett mellom sine servere og aksessnodene, eller han kan leie tilknytning og IP-transport av en annen ISP. IP-nettene til ISP-ene utgjør i praksis Internett backbone i Norge.

En peering-avtale er en avtale om trafikkutveksling uten gjensidig fakturering av kostnader. Peering er en vanlig form for samtrafikk mellom ISPer. Økende grad av direkte samtrafikk mellom norske ISPer, utenom NIX

Oppgave 4:

Hva menes med Round Trip Time (RTT)? Hvilke faktorer påvirker denne?

RTTP er et mål på forsinkelse mellom to maskiner som er den tiden som kreves for en pakke eller datagram å reise fra kilden til målet og tilbake. RTT inkluderer den tiden som kreves for destinasjonen å behandle meldingen fra kilden og til å generere et svar. RTT brukes av enkelte ruting algoritmer til hjelp i beregning av optimale ruter. I de fleste pakkesvitsjing nettverk, forsinkelser varierer som følge av overbelastning. Dermed mål på rundtur ganger gir vanligvis gjennomsnitt, noe som kan ha høy standard avvik.

Oppgave 5:

Hvorfor bruker vi en lagdelt arkitektur innen datakommunikasjon? Hva er OSI-modellen?

Fordi kommunikasjon er en kompleks prosess:

- *lagdeling deler prosessen i mindre deler som er enklere å håndtere*
- *lagdeling forenkler vedlikehold og oppdatering av systemet*
    - *endringer i implementasjonen av et lags tjenester er ikke merkbart for resten av systemet*

OSI-modellen (husk at denne modellen er kun en teoretisk modell):

Hvert lag tar data fra laget over og

- ***legger på en "header" med informasjon og får således en ny dataenhet***

- ***videresender dataenheten til laget under***

Lag 7 - Applikasjon

Lag 7 tilbyr et inngangspunkt for programmer, slik som nettlesere og e-post-systemer slik at disse får tilgang til nettverkstjenester. Dette laget representerer applikasjonprogrammeringsgrensesnitt (API) slik at utviklere kan bruke laget for å utføre nettverksfunksjoner når applikasjoner bygges. FTP, HTTP, SMTP, Telnet, POP3, IMAP, SSL, IMAP, SNMP, DNS. Kommuniserer til transportlaget via socket (portnummer)

Lag 6 – Presentasjon

Lag 6 Oversetter data mellom forskjellige datasystemer på et nettverk. Presentasjonslaget oversetter applikasjonslagets datasyntaks til en felles transportsyntaks egnet til å sendes over nettverket. Når data når frem til målsystemet, oversetter presentasjonslaget hos målsystemet datastrømmen til sin egen syntaks igjen. Dersom to maskiner som kommuniserer har forskjellig dataformat, kan presentasjonslaget konvertere begges format til et felles format. Det er på dette laget at komprimering og de-komprimering foregår, samt kryptering og de-kryptering.

Lag 5 – Sesjon

Lag 5 muliggjør at to applikasjoner kan opprette en vedvarende kommunikasjonsforbindelse(en sesjon). Dette laget sørger for at både sender og mottaker er klare til å kommunisere med hverandre. Dette laget oppretter også kontrollpunkter for å sørge for at kommunikasjonene kan gjennopptas ved forstyrrelser.

Lag 4 - Transport

Lag 4 sørger for at pakker blir levert i den tilstanden de blir sendt og ikke er forandret, tapt eller duplisert. På sendesiden er dette laget ansvarlig for å bryte ned store pakker i mindre pakker for sending på nettverket. På mottakersiden er dette laget ansvarlig for å gjenoppbygge større pakker utifra de mindre, slik at de kan sendes videre til Sesjons-laget. UDP, TCP. Kommuniserer via protokollene 17 i udp og 6 i tcp. En pakke i transportlaget blir kalt for ***Segment.***

Lag 3 - Nettverk

Lag 3 er det eneste laget som benytter «logical networking» og kan flytte pakker mellom forskjellige nettverk. . På Internett sørger laget for at informasjonen havner hos rett maskin. Nettverkslaget må også passe på at dataene blir sendt den mest effektive veien dersom det finnes flere alternativ.IP, Ruting, ICMP. Pakkene i dette laget er kjent som ***Datagram.***

Lag 2 – Datalink (link er kommunikasjonskanalen, for eksempel kabla linker eller trådløse linker)

Lag 2 sørger for en feilfri overførsel av datarammer (frames) mellom datamaskiner igjennom det fysiske laget, lag 1. MAC (media access control) adressen til ett nettverkskort befinner seg i dette laget, og legges til pakken for å skape en ramme (frame). I OSI modellens referansebetydning, er en ramme (frame) en elektronisk konvolutt av informasjon som inkluderer pakken og annen informasjon som legges til av de sju lagene i OSI-modellen. Data-link laget er ansvarlig for å bestemme når rammen skal sendes på nettverket, og så videresende data til det fysiske laget (lag 1). Data sendes fra Data-link laget til det fysiske laget (lag 1) som en strøm av 1-tall og 0. Ethernet, PPP, ATM. Stikkord her er altså ***Rammer***

Lag 1 - Fysisk

Lag 1 oppretter det fysiske grensesnittet og mekanismer for å plassere en rå strøm av databits i nettverkskablene. Ettersom hver «bit» med informasjon mottas fra data-link laget, omgjør det fysiske laget datastrømmen til et passende format og sender det ut på nettverket. På et tråd-nettverk blir hver «bit» omgjort til et elektronisk signal, på fiberoptisk nettverk vil hver «bit» gjøres om til et lyssignal. Kobber, fiber

Application Layer

Referanse: Kap 7

Prinsipper og begreper

Oppgave 6:

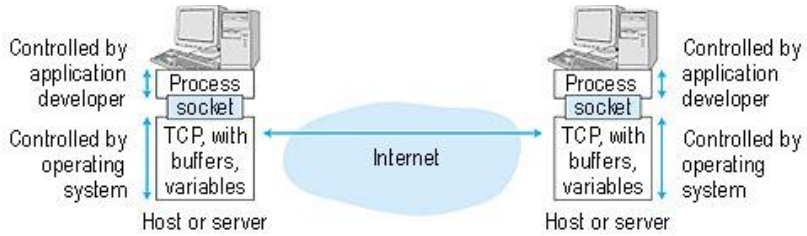
Gi en kort beskrivelse av hva som menes med *klient (client)* og *tjener (server).*

Klient betyr at det er en applikasjon eller system som får tilgang til eksternt tjeneste fra f.eks

Serveren.

Serveren tilbys tjeneste til klienten ved at den mottar forespørsel og gir data eller svar tilbake.

Oppgave 7:



Forklar hva vi mener med en *socket.*

Socket betyr generelt at vi oppretter et endepunkt for kommunikasjon, at to eller flere prosesser kan kommunisere ved å opprette socket og sende meldinger til hverandre.

Oppgave 8:

Hva mener vi med begrepet *Services* når vi snakker om at applikasjons-laget krever/ønsker *Services* fra transportlaget, og hvilke 3 klasser eller dimensjoner av *Services* snakker vi som regel om?

Services vil si at det tilbys tjeneste til et lag høyere enn seg og tjenestene forteller mer hva de gjør enn hva som er inneholdet. Typisk grensesnitt(interface).

Oppgave 9:

Forklar kort tjenestene (*Services)* som tilbys applikasjons-laget fra transportlagsprotokollene UDP og TCP. Viktig å få frem ulikhetene i tjenestetilbudet.

TCP: forbindelsesorientert transporttjeneste: tilbys at det blir opprettet et fysisk forbindelse mellom dem hvor man overfører data til akkurat den man har opprettet forbindelse med

Tilbys typisk pålitelig overføring av data.

UDP: forbindelsesløs transporttjeneste: data blir fragmentert i pakker hvor hver pakke får sin destinasjonsaddresse for hvor de skal bli sendt ut og ruterne bestemmer hvilke vei de skal følge, det blir ikke opprettet noe forbindelse noe som medfører at det er en "best effort" overføring.

Tilbys f.eks RPC, Real time transport,

Oppgave 10:

Hva menes med at en protokoll er *stateless (uten tilstand)*?

Stateless vil si at det er en kommunikasjonsprotokoll som behandler hver forespørsel sopm en avhengig overføring dvs at serveren ikke holder informasjon om tidligere client forespørsel.

Hva menes når vi sier at en forbindelse er *persistent* eller *nonpersistent* (vedvarende eller ikke vedvarende)?

Applikasjonslags Protokoller

HTTP

Oppgave 11:

Beskriv kort HTTP-protokollen. Hva kjennertegner den (ASCII-basert, persistent/ikke persistent, state/stateless etc.) ?

http er webapplikasjonslagets protokoll hvor et eksempel er client/server modellen. http

Er stateless hvor den ikke holder informasjon over tidligere forespørsler. Den har to typer http meldinger, forespørsel og svar. Meldingene er basert på ASCII for å være leselig format.

Hvilken tansport-lags protokoll benytter HTTP ?

TCP

Hvorfor bruker vi *chaching* i HTTP og hvordan fungerer *chaching* –mekanismen ?

Chaching bruk for at man skal gjøre jobben til browseren enklere, man bruker å mellomlagre en siden under en temp mappe lokalt på disken, når man da igjen skal gjøre en søk for oppdatering av siden så vil browseren gjøre en forespørsel til web server om det har blitt gjort noen endring på siden. Har ikke det skjedd så vil browseren bruke siden som allerede ligger lokalt på pc..

Hva er *cookies* ?

Det er en informasjonskapsel, denne kapselen inneholder informasjon om brukeren på siden og den kan også benyttes ved at man handler over nett, cookies vil kunne holde på informasjon om hva som ligger i handelkurven.

FTP

Oppgave 12:

Hva menes med at informasjon sendes *out-of-band* ?

Beskriv kort FTP-protokollen ?

FTP-protokollen sørger for at man har en forbindelse mellom host og server, man seder alt ukryptert og den husker fra gang til gang hva som blir brukt om for eksempel passord og brukernavn.

SMTP

Oppgave 13:

Hvilke tre hovedkomponenter består et Elekronisk post system av ?

De tre hovedkomponentene er meldingsformatet, email addresse og brukeragenten.

Hvilken transportlags tjeneste (service) benytter SMTP seg av ?

Transportslagstjenesten som benytter seg av SMTP er TCP

Beskriv kort SMTP (ASCII/Binær, state/stateless, etc.) ?

SMTP er push, ASCII basert, : Kan sende flere objekter i samme responsmelding. SMTP har state

Hva kan du si om SMTP i forhold til HTTP (ulikheter, likheter) ?

http enkapsler hver objekt av meldinger i sitt eget responsmeldinger, mens SMTP legger all meldingsobjekter i en melding.

SMTP trenger hver melding inkludert kropp for hver melding må være 7 bit ASCII og hvis det inneholder binært, så er meldingen enkodet med 7 bit ASCII.

http henter info fra server mens SMTP sender mail server og dytter filen til mottagerens mailserver.

Hva er *MIME*, og hvorfor trenger vi MIME ?

Gi eksempler på MIME-typer.

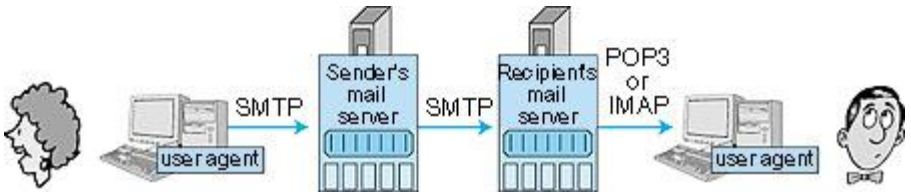
Du kan ikke sende bilder i epost av seg selv siden epost er en ren ASCII – basert tjeneste (7 bits ASCII) bilder og andre vedlegg sendes vha MIME..

Multipurpose Internet Mail Extension..

→ *Bilder, video, dokumenter i ulike format, HTML..*

POP3 og IMAP

Oppgave 14:



Se på figuren over. Hvorfor kan vi ikke benytte SMTP hele veien igjennom (fra avsender til mottaker)?

SMTP er push basert protokoll så den tar meldinger og kobler til et eksternt server til å overføre meldingene. Vi vil gjerne at meldingene blir lagret i mailboksen og må fortsette å lagre i mail transfer agent og pga brukeragaenten ikke er på nettet når SMTP prøver å overføre meldingen.

Gi en kort beskrivelse av POP3-protokollen (ASCII/Bbinær, state/stateless etc.) ?

Hva er den/de viktigeste ulikhetene mellom POP3 og IMAP?

For en bruker som er mye på farten, og gjerne benytter flere maskiner som han ønsker å ha tilgang til sin e-post fra, hvilken e-post løsning er best for han /hun?

POP3 er et enklere protokoll som støtter mindre tillegg og mindre sikker typisk meldinger. Mail er da lastet til brukeragentens data eller pc enn i mailserver. Dette gjør det vanskelig for brukeren og det er vanskelig å lese mail i flere pc'er og hvis pc'en kræsjer vil den miste all email og data.

IMAP: mail lagret hos server hvor IMAP dvs i en server har lagret data om personens mail i en typisk mailbox.(brukernavn og passord må oppgis).

Webmail er en bedre løsning for da bruker den web som et grensesnitt for sending og mottak av meldinger.

DNS

Oppgave 15:

Hva er oppgaven til en DNS-server?

Oppgaven til DNS er hierarkisk domenebasert navn og distribuert databasesystem for implementering av navneskjema, kort sagt brukt for å mappe hostnavn til ip addresse.

Hvilke tre typer DNS-servere snakker vi om, og hva skiller disse?

Root-DNS, Authorative DNS og non-authoritative DNS

- ➔ *Authoritative er sjefs DNS-en som kontaktes når folk utenfra vil ha hjelp..*
  - *Her ligger for eksempel hials.no*
- ➔ *Non-authoritative DNS*
  - *Under DNS-er.. som regel den første DNS-en du tar kontakt med når du vil ut på nettet.. en local DNS…*
  - *Kan som regel være at det bare er en DNS server i et nettverk.. og denne er den autoritative..*

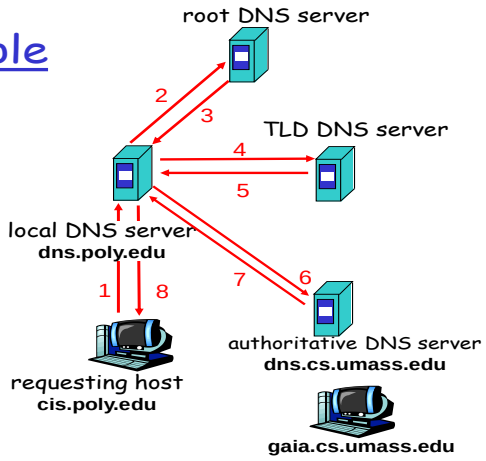
Forklar hva som menes med *iterativ* og *recursive* oppslag (queries)? Vis gjerne med tegninger/eksempel.

- *Iterative oppslag er når man løper nedover en liste og spør folk om noe.. får svar tilbake "prøv her".. evaluerer.. så prøver igjen.. og igjen.. til man kommer nærmere.. og man kommer frem..*
  - *Vi kan si at iterative oppslag er når en DNS står i sentrum og spør root om hvor et sted er.. som så sier tilbake at du kan prøve "her".. DNS som spurte i utgangspunktet spør vedlagt adresse.. får svar tilbake prøv "her" osv..*
  - *Recursive oppslag er når en lokal DNS spør root.. root spør direkte .no-domenet, .no-domenet spør uninett osv.. og når man er nådt målet, sendes svaret tilbake til den lokale DNS-en som spurte i utgangspunktet. Svaret går tilbake samme vei som den tok i utgangspunktet.*

DNS name resolution example

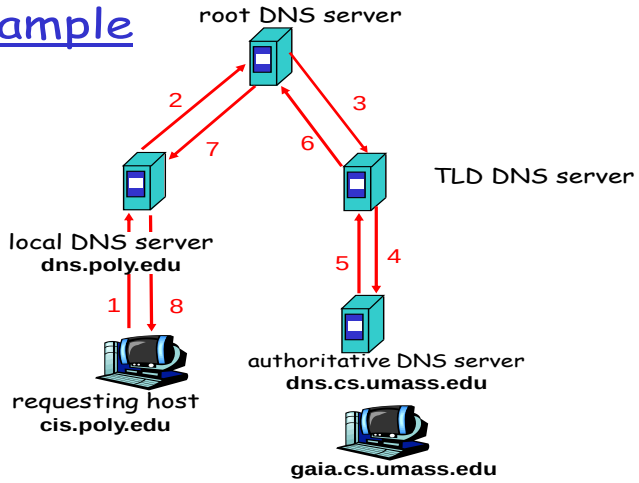
- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:
- puts burden of name resolution on contacted name server
- heavy load?



Oppgave 16:

En DNS resource records (RR) består av følgende felter: (Name, Value, Type, TTL). Type feltet kan være A, NS, CNAME og MX. Hva angir disse?

A – det er å gjøre om en spesifikk url adresse til en ip-adresse (for eksempel www.vg.nowww.vg.no ip)
NS – får tilbake informasjon om vg.no-domenet. Dette kan være epost osv.
CNAME – Cronical name – for å slippe å skrive en lang url, for eksempel db.no -> dagbladet.no
MX – mail -> for eksempel stud.hials.no

P2P applikasjoner

Oppgave 17:

Hva kjennetegner P2P (Pee-to-peer) applikasjoner? Hvilke mekanismer bruker disse for å gi mulighet til søk og utveksling? Nevn eksempler på typiske teknologier som brukes i dag.

Bearshare, frostwire, limewire → gnutellanettverket

Napster – sentralisert fildelingstjeneste, sendte info til server om at "jeg deler sånn og sånn"

- *Funker på samme måte som dagens IRC-servere*
- *Feilen var at saksøkere kunne henvende seg et sted og knuse hele nettverket → de som drifter serverne*

Gnutellanettverket funker slik at du har super peers.. (klientene har sånn funksjonalitet).. som har informasjon om hvem som har hva.. en sentralisert server som du i utgangspunktet spør hvor du kan finne en super peer som du kan søke gjennom… bruker flood-queries.. blir stoppet etter kort tid for ikke å skape mye trafikk på nettet..

Bittorrent bruker trackere der du henter ned info om hvor du kan koble deg på for å få tilgang til å laste ned/opp filen du er ute etter..

- *Målet er å laste opp filen 1 gang (i praksis veldig annerledes.. ratio) andre personer som kobler seg på, laster ned litt.. for så å dele ut det de har lastet ned mens de laster ned nye deler av filen. Teknologien er slik at du i utgangspunktet kobler deg opp mot folk som har samme båndbredde som deg.. så om du struper båndbredden opp, struper du også båndbredden ned..*
- *Torrenter dør... når ingen laster ned/opp, får ingen andre heller lastet ned filen.. altså avhengig av at folk seeder (nesten).. "availability" bør være over 1.. du trenger ikke ha seedere.. men så lenge tilgangen til filen er under 1.. betyr dette at filen ikke ligger fullstendig ute.. da vil du aldri få hele filen ned ettersom den ikke ligger ute..*

1. Aplikasjonslaget

- *a) Forklar kort hva DNS er, og hvilken anvendelse det har i Internet*

  Domain Name System er et hierarkisk navngivingsskjema brukt i Internett. Systemet er implementert som et distribuert databasesystem, med servere over hele verden. Ved å sende querry til en DNS-server kan en prosess mappe fra et Internet domenenavn til IPadressen som benyttes for å kommunisere med det domenet. Topnivådomenene er enten generiske (.com .edu etc) eller nasjonale (.no .us etc) . Bladnodene i navnegrafen er domener som ikke har noen underdomener. Ethvert domene har 'resource records' eks MX, A osv som de fleste vil kjenne igjen fra parameter til og resultater av spørringer med kommandoen dig.

- *b) Hva er SMTP og MIME, og hvilken Internet-applikasjon benytter disse?*

  De fleste epostsystemer i verden følger RFCstandardene 2821 og 2822 hvor meldinger sendt bryker ASCII-headere til å definere meldingsegenskaper. Ved hjelp av MIME, *Multipurpose Internet Mail Extensions* kan mange ulike typer innhold overføres. Dette muliggjør å følge overnevnte RFC samtidig som body i meldingen strukturerers og ikke-ASCII meldinger får std. regler for koding. SMTP, *Simple Mail Transsfer Protocol,* benyttes til å sende meldingene ved å opprette en TCP-forbindelse mellom src og dest host, og direkte levere eposten over denne forbindelsen.

- *c) Hva består forskjellen mellom server-side og client-side scripts i WWW i?*

Scripts brukes generelt til å konstruere dynamiske webdokumenter. CGI, PHP, JSP og ASP løser problemet med å håndtere forms og interaksjoner med databaser på serveren. De kan alle motta innkommende info fra forms, slå opp infoen i en eller flere databaser, og generere HTMLsider med resultatet. Det ingen av dem imidlertid kan gjøre, er å reagere på musbevegelse eller samhandle med brukeren direkte. Til dette formålet trengs script embedded i HTMLsider som eksekveres på klientmaskinen istedenfor på servermaskinen. Det vanligste er bruk av Javascript.

- *d) Nevn noen vanlige Internet-applikasjoner som benytter TCP, og skisser hendelsesforløpet under en slik sesjon.*

  Epost og www. Eks;
  www-klientside: browser avgjør url og spør DNS om IPadressen. browser oppretter en TCPforbindelse til port 80 på den IPadressen den fikk fra DNS. Deretter sendes det request om aktuell fil, og serveren overfører filen. TCPforbindelsen tas så ned og filen vises frem til brukeren. www-serverside: serveren aksepterer TCPforbindelsen fra klienten(browseren) , mottar filnavnet og henter filen fra disk (o.l.) og returnerer filen til klienten før forbindelsen frigjøres igjen.

- *e) Hva er WAP? HVilke utfordringer medfører WAP i forhold til vanlig Internett?*

  Wireless Application Protocol, WAP, er bygd på ideen om å bruke den eksisterende digitale trådløse infrastrukturen til webapplikasjoner. Barnesykdommene til WAP er preget av liten skjerm til å vise frem applikasjonen, og lav båndbredde. Følgelig har utbredelsen gått tregt siden folk flest ikke foretrekker å aksessere nettet fra små mobilskjermer med lav oppløsning og tikkende tellerskritt. Protokollen er optimert for lav-båndbredde forbindelser med trådløse apprater som har treg CPU, lite minne og liten skjerm; noe som er klart forskjellig fra dagens "allemannseide" PC'er. Man må derfor tenke praktisk nytte (og minimalisme) fremfor fancy design.

- *f) Hvilke hensyn må man ta i sammenheng med multimedia applikasjoner over Internett?*

  Siden audio er mindre båndbreddekrevende enn video har utviklingen kommet noe lenger her. Båndbreddebegrensninger og QoSrelaterte parameter som max delay og jitter er særdeles relevante. Dette har ledet til videreutvikling av komprimeringsteknikker så vel som standarder for overføring av slike data. Det kreves også en god del buffringskapasitet for å kunne levere en "smooth" realtime tjeneste hos klienten. Ettersom PC'er blir stadig kraftigere og de fleste hjem har Internett-tilgang , forventer man at kvaliteten på webaplikasjonene skal leve opp til den kvaliteten man opplever lokalt.

2. Oppgaver fra Tanenbaum kap. 7

1.1.1 7-5

DNS er 'indempotent' ; operasjoner kan gjentas uten skade. Når en prosess gjør en DNS request startes en timer. Hvis denne utløper, utføres bare requesten på nytt uten bivirkninger.

1.1.2 7-7

Ja. Se fig 7-3 for et eksempel på duplikat IP-adresse. Husk at en IP-adresse består av et nettverksnummer og et hostnummer. Hvis en maskin har to ethernetkort, kan den være del av to separate nettverk, og således trenge to IP-adresser.

1.1.3 7-8

Det er mulig. eks www.large-bank.com og www.large-bank.ny.us Evs, det er ganske vanlig å ha entry både under .com og et nasjonalt domene.

1.1.4 7-16

Ja, bruk subtypen message/external-body og send url til filen istedenfor selve filen.

1.1.5 7-20

Nei. POP3 programmet rører ikke egentlig remote mailbox. Den sender kommandoer til en POP3 daemon på mailserveren. Så lenge daemonen forstår mailboxformatet kan det fungere. Dvs at en mailserver kan endres fra ett format til et annet over natten uten at kundene informeres, forutsatt at POP3 damonen samtidig endres slik at den forstår det nye formatet.

1.1.6 7-21

Lagring av brukeres epost tar opp diskplass, noe som koster penger. Denne faktoren argumenterer for bruk av POP3. På den annen side kan ISP ta betalt for diskplass ut over noen få megabyte og dermed gjøre epostsystemet til en pengegruve. Det siste alternativet taler for IMAP for å oppmuntre brukere til å beholde epost på serveren (og betale for diskplassen).

1.1.7 7-22

Webmail bruker ingen av dem, men likner ganske mye på IMAP fordi begge tillater en remote klient å undersøke og håndtere remote mailboxes. I kontrast sender POP3 mailboxen til klienten for prosessering der.

1.1.8 7-28

DNS-navn kan ikke ende med et siffer, så det er ingen tvetydighet involvert.

1.1.9 7-48

Det tar 50 msec å få en pausekommando til serveren, og på den tiden vil 6250B ankomme, slik at low-water mark bør være godt over dette tallet, trolig rundt 50000 for å være på den sikre siden. På samme måte bør high-water mark være minst 6250B fra toppen, gjerne 50000 for å ha god margin også i denne enden.

1.1.10 7-49

Det medfører ekstra forsinkelse (delay). Ifølge 'strait-forward' skjemaet kan den første pakken sendes etter at 5 msec har forløpt. I dette skjemaet må systenet vente hele 10 msec før det kan sende sampler for de første 5 msec.

1.1.11 7-52

JA. En feil i en I-frame vil forårsake feil i rekonstruksjonen av påfølgende P-frames og B-frames. Faktisk vil feilen propageres helt til den neste I-frame!

Transport Layer

Referanse: Kap 3

Prinsipper og begreper

Oppgave 18:

Hva er hovedoppgaven til Transportlaget?

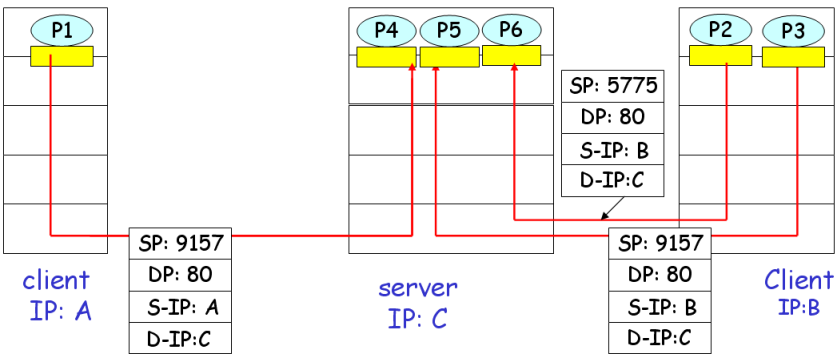
Hovedoppgaven til Transportlaget er å overføre data fra maskin til maskin.

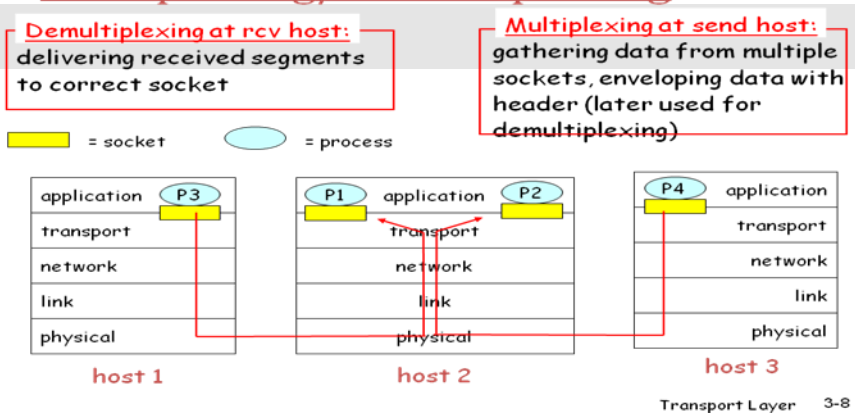
Hva er en av hovedforskjellene mellom netverks- og transportlaget?

Hoved forskjellen mellom transport og nettverkslaget er at nettverkslaget tilbyr ende til ende pakkesending ved å bruke datagram eller virtuell krets, transportlaget overfører data fra ressurs til destinasjon maskin med robusthet.

Oppgave 19:

Forklar hva som menes med *Multiplexing/demultiplexing* på transportslaget?





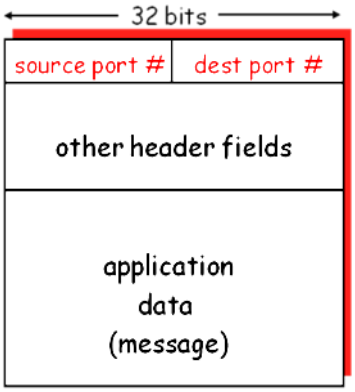
Multiplexing: Samle data fra flere applikasjons-prosesser, legge til header og sende til nettverks laget. Demultiplexing: Levere mottatte segmenter til riktig applikasjonslags prosess.

Hvordan klarer transportlaget å skille mellom applikasjonslags-prosesser under multiplexing/demultiplexing?

- *Multiplexing: Tar datastrømmer fra forskjellige prosesser og sender dem videre ned i stakken som en datastrøm via IP*
- *Demultiplexing: prosessen ved å splitte opp datastrømmen til flere datastrømmer og guide til riktige prosesser..*
- *Skilles vha portnummer..* → *dette er grunnen til at flere programmer/prosesser/nettverksstrømmer ikke kan bruke den samme porten samtidig.*

How demultiplexing works

- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries 1 transport-layer segment
  - each segment has source, destination port number
- host uses IP addresses & port numbers to direct segment to appropriate socket



TCP/UDP segment format

Connectionless demultiplexing

- Create sockets with port numbers:

```
DatagramSocket mySocket1 = new
   DatagramSocket(12534);
DatagramSocket mySocket2 = new
   DatagramSocket(12535);
```

- UDP socket identified by two-tuple:

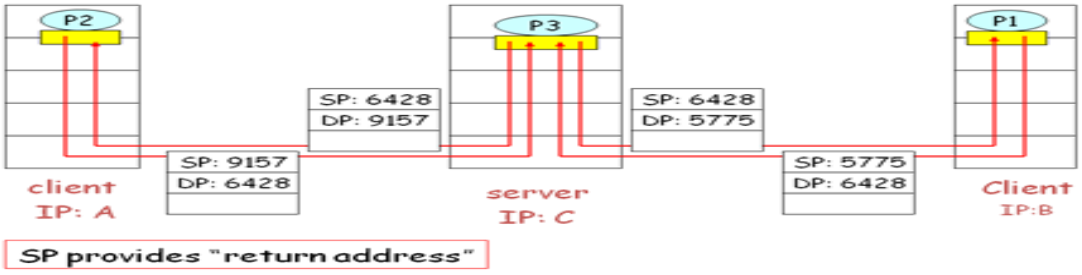
(dest IP address, dest port number)

- When host receives UDP segment:
  - checks destination port number in segment
  - directs UDP segment to socket with that port number
- IP datagrams with different source IP addresses and/or source port numbers directed to same socket

→*Demux = demultiplexing, cont = continued*

Connectionless demux (cont)

```
DatagramSocket serverSocket = new DatagramSocket(6428);
```

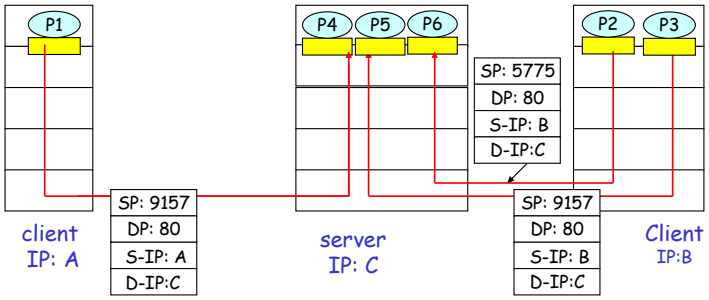


SP provides "return address"

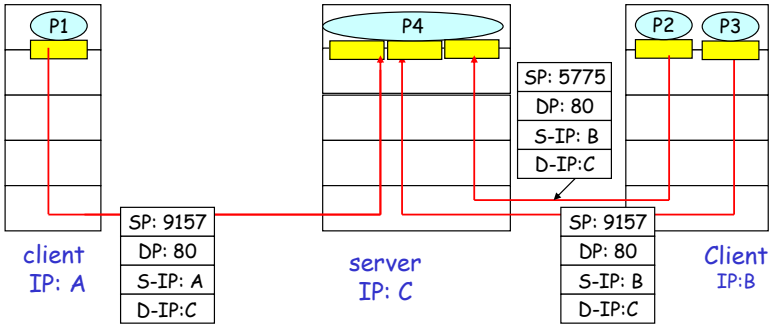
Connection-oriented demux

- TCP socket identified by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number
- recv host uses all four values to direct segment to appropriate socket
- Server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple
- Web servers have different sockets for each connecting client
  - non-persistent HTTP will have different socket for each request

Connection-oriented demux (cont)



Connection-oriented demux: Threaded Web Server



Oppgave 20:

Hva menes med en *connectionless- (forbindelsesløs-)* og en *connection oriented (forbindelsesorientert)* protokoll?

Connectionless: Det blir ikke opprettet en forbindelse mellom avsender og mottaker før data utveksles. Eksempel: UDP.

UDP: User Datagram Protocol [RFC 768]

- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - lost
  - delivered out of order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

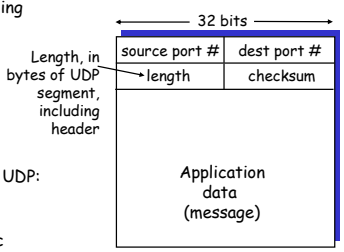
Why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired

Transport Layer 3-16

UDP: more

- often used for streaming multimedia apps
  - loss tolerant
  - rate sensitive
- other UDP uses
  - DNS
  - SNMP
- reliable transfer over UDP: add reliability at application layer
  - application-specific error recovery!



UDP segment format

Transport Layer 3-1

UDP checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted segment

Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?* More later ....

Transport Layer 3-18

Internet Checksum Example

- Note
  - When adding numbers, a carryout from the most significant bit needs to be added to the result
- Example: add two 16-bit integers

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	(1)	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Transport Layer 3-19

Her kan en avsender sende en bråte pakker uten i det hele tatt å være sikker på at en mottaker finnes. Avsender får heller aldri en beskjed om dette. Kan brukes til å ta opp all båndbredde i et nettverk dersom du vil kjøre et DDOS-angrep. Switsjer og rutere får så mye å gjøre med UDP trafikken at de ikke får tid til å gjøre noe annet..

Connection Oriented: En forbindelse opprettes i forkant (handshake), før data utveksles. Når overføringen av data er ferdig, lukkes/termineres forbindelsen. Eksempel: TCP.

Oppgave 21:

Hvilke utfordringer/problemer må løses for at vi skal kunne tilby en pålitelig (reliable) kommunikasjon, og hvilke mekanismer kan vi benytte for å løse problemet?

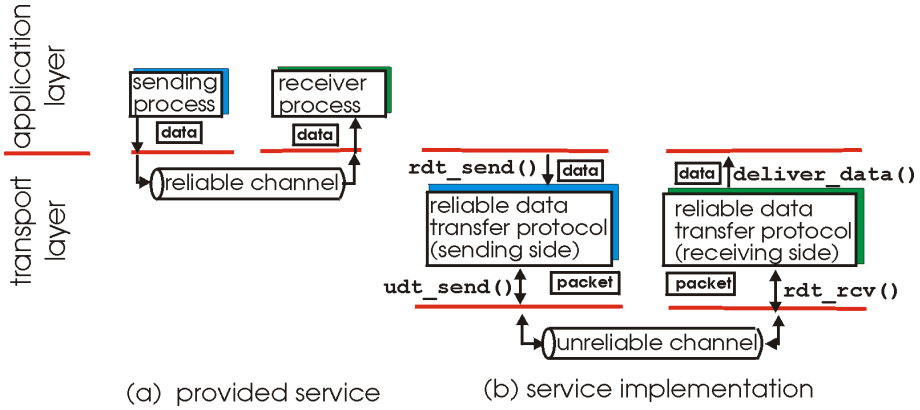
1) *Data i et segment kan bli endret/ødelagt på vei fra sender til mottaker. Løsning: Checksum for å detektere feil, ACK/NAK for å "rapportere" feil.*

2) *ACK/NAK kan også bli korrupt. Løsning: Retransmisjon ved mottak av korrupt ACK/NAK, samt sekvensnummerering.*

3) *Data kan gå tapt i nettet. Løsning: Timeout.*

Principles of Reliable data transfer

- important in app., transport, link layers
- top-10 list of important networking topics!



(a) provided service (b) service implementation

- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
  - checksum to detect bit errors
- *the* question: how to recover from errors:
  - *acknowledgements (ACKs):* receiver explicitly tells sender that pkt received OK
  - *negative acknowledgements (NAKs):* receiver explicitly tells sender that pkt had errors
  - sender retransmits pkt on receipt of NAK
- new mechanisms in `rdt2.0` (beyond `rdt1.0`):
  - error detection
  - receiver feedback: control msgs (ACK,NAK) rcvr->sender

rdt2.0 has a fatal flaw!

What happens if ACK/NAK corrupted?

- sender doesn't know what happened at receiver!
- can't just retransmit: possible duplicate

Handling duplicates:

- sender retransmits current pkt if ACK/NAK garbled
- sender adds *sequence number* to each pkt
- receiver discards (doesn't deliver up) duplicate pkt

stop and wait
Sender sends one packet, then waits for receiver response

rdt2.2: a NAK-free protocol

- same functionality as rdt2.1, using ACKs only
- instead of NAK, receiver sends ACK for last pkt received OK
  - receiver must *explicitly* include seq # of pkt being ACKed
- duplicate ACK at sender results in same action as NAK: *retransmit current pkt*

RDT =reliable data transfer

Gitt løsningene over, hvilke problemer står vi igjen med i en klassisk *Stop-and-Wait* løsning (sender data, venter på ACK), og hvordan kan vi løse dette problemet?

Svært dårlig utnyttelse av båndbredde, siden vi må vente på ACK før neste datasegment kan sendes.
Løsning: Pipelining. M.a.o. sende flere segmenter før vi venter på å motta ACKer.

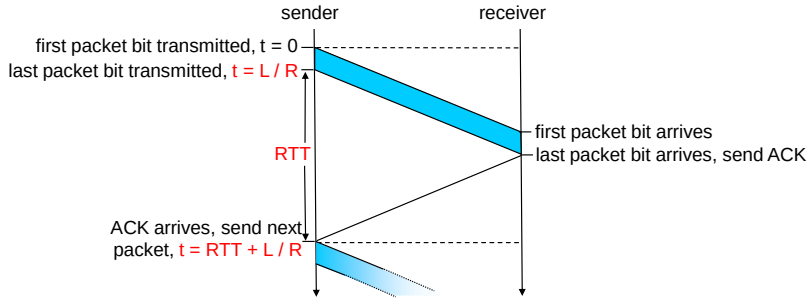
Standard størrelse – 1500 bytes, payload 1460..

RTT vil ha mye å si i et slikt scenario..

Pipelining.. sender mange pakker samtidig uten at du har fått ACK..

Ackumulativ ACK.. en ACK for mange pakker

rdt3.0: stop-and-wait operation



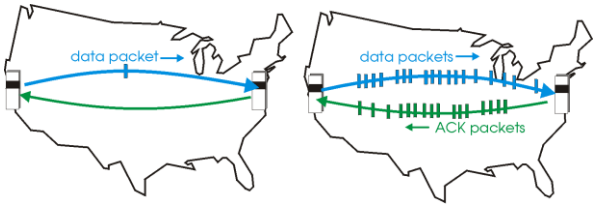
$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Transport Layer 3-42

Pipelined protocols

Pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged pkts

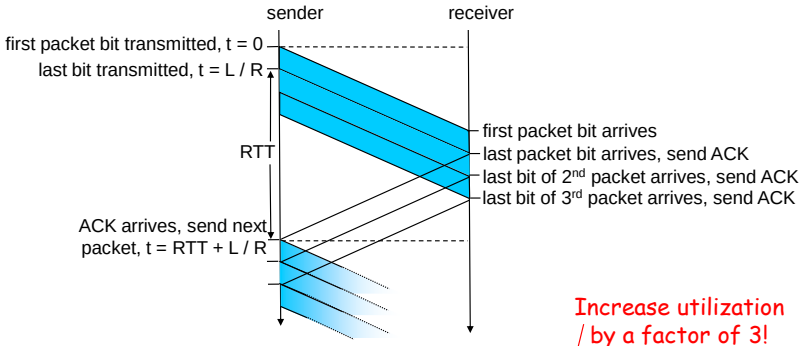
- range of sequence numbers must be increased
- buffering at sender and/or receiver



(a) a stop-and-wait protocol in operation (b) a pipelined protocol in operation

- Two generic forms of pipelined protocols: *go-Back-N, selective repeat*

Pipelining: increased utilization



Increase utilization by a factor of 3!

$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Pipelining Protocols

Go-back-N: big picture:

- Sender can have up to N unacked packets in pipeline
- Rcvr only sends cumulative acks
  - Doesn't ack packet if there's a gap
- Sender has timer for oldest unacked packet
  - If timer expires, retransmit all unacked packets

Selective Repeat: big pic

- Sender can have up to N unacked packets in pipeline
- Rcvr acks individual packets
- Sender maintains timer for each unacked packet
  - When timer expires, retransmit only unack packet

Oppgave 22:

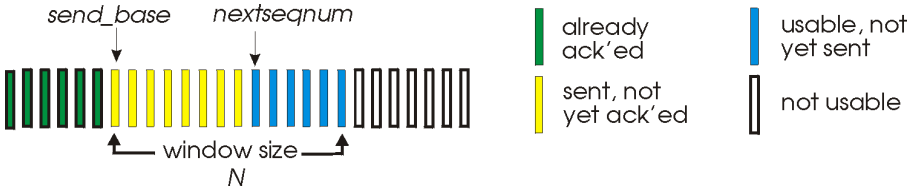
Forklar hovedprinsippene i Go-Back-N (GBN)?

➔ *Sender en bråte pakker.. får du ikke ACK på en pakke oppi der, så sendes alle pakker fra og med manglende ACK på nytt.. – i bruk i TCP stort sett*

Go-Back-N

Sender:

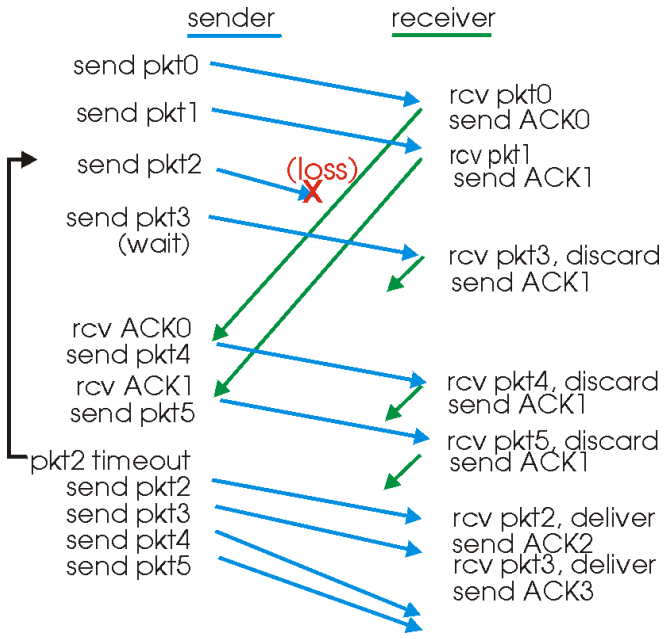
- k-bit seq # in pkt header
- "window" of up to N, consecutive unack'ed pkts allowed



- ACK(n): ACKs all pkts up to, including seq # n - "cumulative ACK"
  - may receive duplicate ACKs (see receiver)
- timer for each in-flight pkt
- *timeout(n):* retransmit pkt n and all higher seq # pkts in window

Transport Layer 3-47

GBN in action



Transport Layer 3-50

Forklar hovedprinsippene i Selective Repeat?

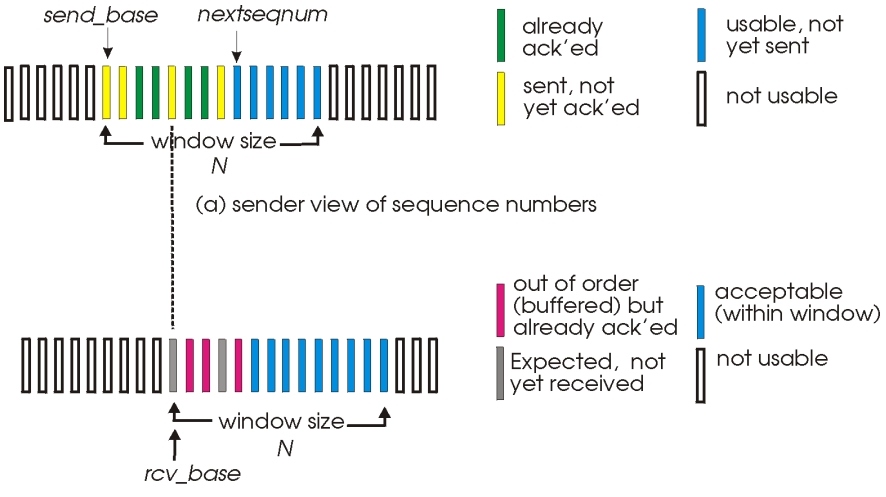
➔ *Ikke i bruk av TCP.. – går på å sende mange pakker.. og så sende pakken som du ikke får ACK på..*

➔ *Må holde timeouts på alle pakker, ressurskrevende, mye overhead*

Selective Repeat

- receiver *individually* acknowledges all correctly received pkts
  - buffers pkts, as needed, for eventual in-order delivery to upper layer
- sender only resends pkts for which ACK not received
  - sender timer for each unACKed pkt
- sender window
  - N consecutive seq #'s
  - again limits seq #s of sent, unACKed pkts

Selective repeat: sender, receiver windows



Selective repeat

sender

data from above :

- if next available seq # in window, send pkt

timeout(n):

- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:

- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

receiver

pkt n in [rcvbase, rcvbase+N-1]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]

- ACK(n)

otherwise:

- ignore

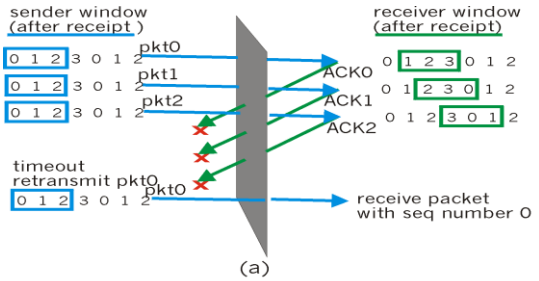
Selective repeat: dilemma

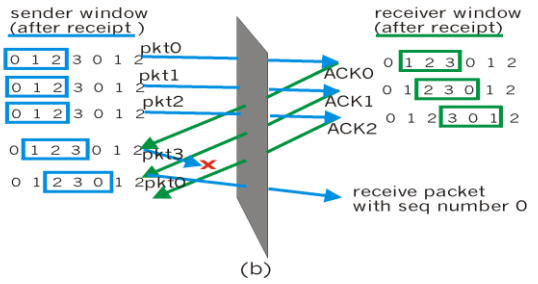
Example:

- seq #'s: 0, 1, 2, 3
- window size=3

- receiver sees no difference in two scenarios!
- incorrectly passes duplicate data as new in (a)

Q: what relationship between seq # size and window size?





1.1.1. UDP

Oppgave 23:

Gi en kort beskrivelse av UDP (connectionless/connection-oriented, flowcontrol etc.)?

UDP er connecti less. Har ingen støtte for flow-control eller congestioncontrol.

- *Har jo checksum felt som brukes til å avgjøre om en pakke er intakt.. forkastes hvis den ikke er det.. gjør ikke noe mer.. programmerer oppe i systemet/applikasjonen må velge om dette skal gjøres noe med.. men som regel trengs ikke noe gjøres.. IP-telefoni osv..*

Hvilke typiske applikasjonslagsprotokoller benytter UDP?

Multimedia applikasjoner og spill.

1.1.2. TCP

Oppgave 24:

Gi noen karakteristika til TCP (connectionless/connection-oriented, state-less etc.)?
Hvilke mekanismer bruker TCP for å oppnå en pålitelig kommunikasjon?

Checksum, sekvensnummerering av segmenter, timeout og forbindelsesorientert. Har en grad av state ettersom den i handshaken oppretter en forbinelse som huskes til forbindelse blir terminert..

Threeway handshake for å si fra at her kommer det pakker..

TCP er et eksempel på en NAK-free protokoll. Hva menes med dette, og hvordan er dette løst i TCP?

Protokollen inneholder ikke NAK. D.v.s. dersom mottaker ikke godkjenner pakken, sendes ikke NAK. Istedenfor sender mottaker en ACK med sekvensnummeret til det segmentet som mottakeren sist mottok og godtok. (NAK = Negative ACK)

Hvis den ikke mottar en pakke.. sendes ikke NAK.. men identisk ACK som matcher den som var sist mottatt korrekt..

Dersom vi da mottar 3 identiske ACK, vet vi at vi mangler en pakke inne i der.. da kan vi sende alle pakker fra den siste på nytt.. (helge tor sier at den kan sende bare manglende pakke.. men hvordan? → selective repeat)

→problem: dersom du mister flere pakker.. og kun mottar ACK fra den siste mottatte pakken.. kan det være at du ikke får vite om de andre pakkene du har tapt? Da må vel alle pakker fra siste mottatte ACK sendes på nytt for å være sikker?

→ mottaker sender ack for hver individuell pakke og timer vedlikeholdes for hver sendt pakke. Dersom timer går til 0, blir en ny pakke sendt. Kan forestille meg at siden alle pakker etter siste mistede pakke ikke vil få noen ack tilbake og dermed vil timeren gå ut.. alle pakker fra og med mistet pakke vil bli sendt på nytt. → virker ikke som om selective repeat er spesielt effektivt her..

- *Selective repeat virker mer logisk til å fungere dersom mottaker sender ack tilbake på alle mottatte pakker.. om en pakke inni strømmen blir mistet eller ack ikke kommer frem, blir resten av pakkene bufret (og ack blir sendt tilhørende riktig pakke), mens timeren på mistet pakke går til 0 og blir dermed sendt på nytt. Når mottaker så mottar riktig pakke, vil ack bli sendt og alt er bra igjen.. håpa oss ☺*

Oppgave 25:

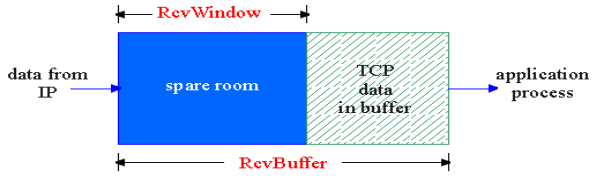
Hva menes med *Flow-control* og hvordan er dette løst i TCP?

Kontroll for å forhindre at sender overfyller bufferet til mottaker ved å sende for mye data for raskt for mottakeren å ta unna.

Løses ved at mottaker i sitt ACK-segment kan angi maks ledig plass i sitt mottaksbuffer, som så avsender benytter for å redusere sitt send-vindu (antall ikke-ACKede segmenter).

TCP Flow Control

- receive side of TCP connection has a receive buffer:

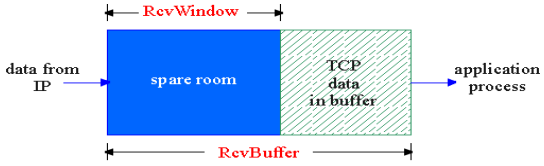


- app process may be slow at reading from buffer

flow control
sender won't overflow receiver's buffer by transmitting too much, too fast

- speed-matching service: matching the send rate to the receiving app's drain rate

TCP Flow control: how it works



(Suppose TCP receiver discards out-of-order segments)

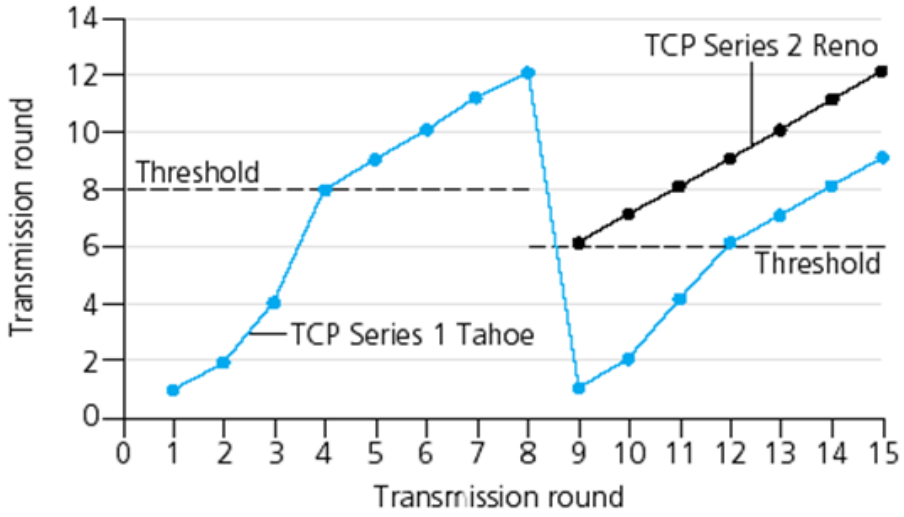
- spare room in buffer

```
= RcvWindow
= RcvBuffer-[LastByteRcvd -
  LastByteRead]
```

- Rcvr advertises spare room by including value of `RcvWindow` in segments
- Sender limits unACKed data to `RcvWindow`
  - guarantees receive buffer doesn't overflow

Oppgave 26:

Hva menes med *Congestion control*, og hvordan er dette løst i TCP (stikkord: probing, slowstart, congestion avoidance…)?



Kontrollmekanisme for å forhindre at nettverket jammes ved at for mye data sendes fra avsender for raskt.

Løses ved at overføringsraten begrenses av Congestion window. Størrelsen på Congwin settes til 1 segment når sendig begynner. Deretter økes størrelsen eksponentiellt (Slow Start) til et gitt threshold-nivå nås. Deretter økes Congwin med 1 til segmenter begynner å gå tapt (loss) (congestion avoidance fasen). Threshold settes til Congwin/2 og Congwin starter på 1 igjen, og økes deretter med 1for hver sending.

→*starter med slow start – øker eksponentielt..*

→*når man når et threshold, øker med 1 for hver gang til man taper.. når man taper (får tre like ACK).. halvere senderate (å miste en pakke er jo sånt som skjer).. får man en timeout (ikke mottar ACK) kan det tyde på noe verre.. da begynner man må nytt med slowstart..*

TCP reliable data transfer

- TCP creates rdt service on top of IP's unreliable service
- Pipelined segments
- Cumulative acks
- TCP uses single retransmission timer
- Retransmissions are triggered by:
  - timeout events
  - duplicate acks
- Initially consider simplified TCP sender:
  - ignore duplicate acks
  - ignore flow control, congestion control

Principles of Congestion Control

Congestion:

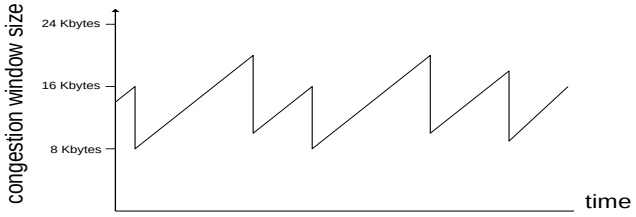
- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!

Viktig å skille på flow control og congestion control
-> flow control ligger i endesystemene
->congestion control ligger på nettverket

TCP congestion control: additive increase, multiplicative decrease

- *Approach:* increase transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **CongWin** by 1 MSS every RTT until loss detected
  - *multiplicative decrease*: cut **CongWin** in half after loss

Saw tooth behavior: probing for bandwidth



TCP Congestion Control: details

- sender limits transmission:
  `LastByteSent-LastByteAcked ≤ CongWin`
- Roughly,

$$\text{rate} = \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

- **CongWin** is dynamic, function of perceived network congestion

How does sender perceive congestion?

- loss event = timeout *or* 3 duplicate acks
- TCP sender reduces rate (**CongWin**) after loss event

three mechanisms:

- AIMD
- slow start
- conservative after timeout events

AIMD = Additive Increase, multiplicative decrease

TCP Slow Start

- When connection begins, **CongWin** = 1 MSS
  - Example: MSS = 500 bytes & RTT = 200 msec
  - initial rate = 20 kbps
- available bandwidth may be >> MSS/RTT
  - desirable to quickly ramp up to respectable rate

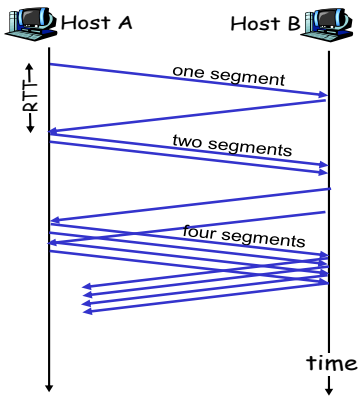
- When connection begins, increase rate exponentially fast until first loss event

Grensen er først satt til 1 segment.

→ vi prøver oss frem litt forsiktig første gangen.. Men øker eksponensielt til første tap.. Da går vi tilbake til sagtaggetingen som på et par slida foran…

TCP Slow Start (more)

- When connection begins, increase rate exponentially until first loss event:
  - double **CongWin** every RTT
  - done by incrementing **CongWin** for every ACK received
- Summary: initial rate is slow but ramps up exponentially fast



Refinement: inferring loss

- After 3 dup ACKs:
  - **CongWin** is cut in half
  - window then grows linearly
- But after timeout event:
  - **CongWin** instead set to 1 MSS;
  - window then grows exponentially
  - to a threshold, then grows linearly

Philosophy:

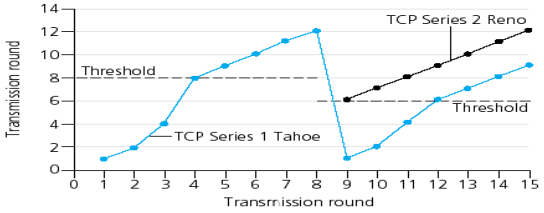
- 3 dup ACKs indicates network capable of delivering some segments
- timeout indicates a "more alarming" congestion scenario

Timeout events er sannsynligvis en mer alvorlig situasjon enn 3 dup acks..

Refinement

Q: When should the exponential increase switch to linear?

A: When **CongWin** gets to 1/2 of its value before timeout.



Implementation:

- Variable Threshold
- At loss event, Threshold is set to 1/2 of CongWin just before loss event

Mellom 1 og 4 har vi mekanismen slow-start
→dersom vi får tilbake 3 identiske acks (tegn på at vi har mistet en pakke) for eksempel mellom 8 og 9 her, senke threshold

Summary: TCP Congestion Control

- When **CongWin** is below **Threshold**, sender in slow-start phase, window grows exponentially.
- When **CongWin** is above **Threshold**, sender is in congestion-avoidance phase, window grows linearly.
- When a triple duplicate ACK occurs, **Threshold** set to **CongWin/2** and **CongWin** set to **Threshold**.
- When timeout occurs, **Threshold** set to **CongWin/2** and **CongWin** is set to 1 MSS.

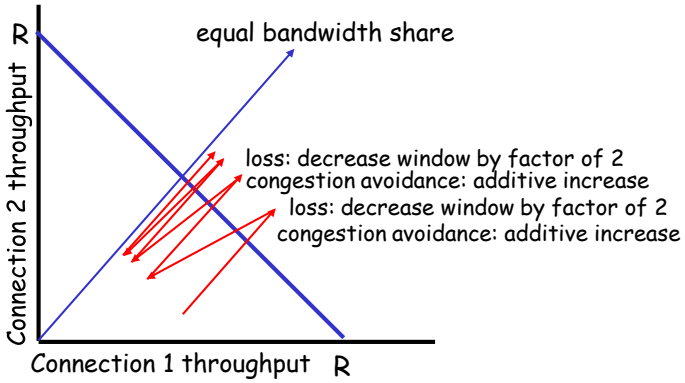
TCP sender congestion control

State	Event	TCP Sender Action	Commentary
Slow Start (SS)	ACK receipt for previously unacked data	CongWin = CongWin + MSS, If (CongWin > Threshold) set state to "Congestion Avoidance"	Resulting in a doubling of CongWin every RTT
Congestion Avoidance (CA)	ACK receipt for previously unacked data	CongWin = CongWin+MSS * (MSS/CongWin)	Additive increase, resulting in increase of CongWin by 1 MSS every RTT
SS or CA	Loss event detected by triple duplicate ACK	Threshold = CongWin/2, CongWin = Threshold, Set state to "Congestion Avoidance"	Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.
SS or CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, Set state to "Slow Start"	Enter slow start
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed

Why is TCP fair?

Two competing sessions:

- Additive increase gives slope of 1, as throughout increases
- multiplicative decrease decreases throughput proportionally



Fordi en PC med høyere båndbredde vil halvere mye mer for hver halvering enn en PC med mindre båndbredde, for så å følge på med additive increase, vil PCene etter hvert utgjevne hverandres båndbredde..

→ *Det finnes veier rundt dette; Siden disse prinsippene gjelder for hver TCP-tilkobling, kan du opprette flere TCP-forbindelser og på denne måten få mer båndbredde.*

→ *Dersom du kjører nedlastinger på en PC og prøver å surfe på en annen, kan nedlastingen for eksempel opprette 9 TCP-tilkoblinger for å laste ned, mens kanskje surfePC har 1… slik vil fairness-prinsippet gi nedlastingsPC 90% av båndbredde, mens surfePC får 10%.*

Fairness (more)

Fairness and UDP

- Multimedia apps often do not use TCP
  - do not want rate throttled by congestion control
- Instead use UDP:
  - pump audio/video at constant rate, tolerate packet loss
- Research area: TCP friendly

Fairness and parallel TCP connections

- nothing prevents app from opening parallel connections between 2 hosts.
- Web browsers do this
- Example: link of rate R supporting 9 connections;
  - new app asks for 1 TCP, gets rate R/10
  - new app asks for 11 TCPs, gets R/2 !

1. Transportlaget

1. *Hvorfor sier vi at transportlaget danner et skille i OSI modellen? Hva er den fundamentale forskjellen på lagene over og under transportlaget?*

   Transportlaget er det første laget med ende til ende kontroll. Lagene fra transportlaget og over finnes typisk bare i endesystemene og ikke i rutere og svitsjer. Over transportlaget finner vi protokoller som understøtter samvirke mellom distribuerte applikasjoner; disse protokollene beskjeftiger seg ikke med selve overføringen av data, men forutsetter at dette ivaretas av underliggende lag. Under transportlaget har vi det fysiske nettet med rammer og ruting. Disse lagene (fysisk, link, nett, transport) har det kollektive ansvaret for overføring av data mellom sender og mottaker, og sørger for at overføringskvaliteten tilsvarer applikasjonenes behov.

2. *Hvilke hovedkrav må transportlaget oppfylle dersom det har det fulle ansvaret for påliteligheten i kommunikasjonen mellom to transport-bruker entiteter?*

   Hvis vi ønsker å oppnå/garantere full pålitelighet i transportlaget kreves følgende: *Pålitelig oppkobling* av forbindelser, *pålitelig overføring* av data, dvs at alle meldinger kommer frem, at ingen meldinger dupliseres, at meldinger ikke bytter rekkefølge og at meldingene ikke inneholder feil (ende-til-ende feilsjekk), *pålitielig nedkobling* av forbindelser uten tap av datapakker (graceful close).

3. *På hvilket grunnlag velges transportprotokoll (eks TCP eller UDP i TCP/IP) for en applikasjon? Når gjøres valget?*

   Valget foretas når man skal implementere en applikasjon, dvs at man velger når det opprettes feks en socket som kan ha parameter UDP eller TCP. Man bør velge den protokollen man tror best kan oppfylle kravene til nettverkskommunikasjon hhv. effektivitet, hastighet, sikkerhet. Ofte ender man med et kompromiss hvor en bruker den protokollen som har flest av de ønskede egenskapene. UDP vil typisk være egnet til å sende mange små pakker, lyd/bilde og interaktiv bruk hvor det ikke er vesentlig at alle pakker kommer frem. TCP er derimot velegnet til feks store filoverføringer.

4. *Beskriv feltene i henholdsvis UDP og TCP headeren. Hvorfor er de så ulike?*

   Se figur side 526 og 537 i Tanenbaum for header layout hhv UDP og TCP. TCP headeren er mer omfattende fordi man trenger å skille mellom ulike typer TPDUer (feks for op- og nedkobling), det trengs sekvensnummer for å sikre at alle bytene leveres i riktig rekkefølge , og det trengs et felt for å gi melding om mottakervinduet.

5. *Transportprotokollen UDP er som IP forbindelsesløs, hvorfor har vi en forbindelsesløs transportprotokoll?*

   Det er behov for valgmuligheter: i noen situasjoner er det bedre/mer effektivt/raskere med en forbindelsesløs transport protokoll. Feilhåndtering etc må da håndteres på laget over (applikasjonen). Ofte skreddersyr man applikasjoner for å få bedre ytelse vha. UDP i situasjoner hvor det sendes mange små pakker, eller tale/video hvor npen pakker kan bli borte uten å ødelegge for resutatet.

6. *Beskriv prinsippet for pålitelig etablering av en transportforbindelse ved hjelp av tre-veis håndtrykk.*

   Se figur side 540 i Tanenbaum. Algoritmen treveis håndtrykk brukes av TCP for å etablere og terminere forbindelser. Håndtryket involverer utveksling av tre beskjeder mellom klient og tjener. Ideen er at de to partene skal enes om et sett parameter for forbindelsen. Ved åpning av forbindelse er det særlig viktig å avtale hvilket sekvensnummer man starter med. I tillegg kan det være QoS-parametre samt andre opsjoner. Først sender klienten en transport protokoll enhet (TPDU) til tjeneren og angir sitt initielle sekvensnummer (>YN) . Tjeneren svarer med en ACK på klientens sekvensnummer samtidig som den angir eget initielle sekvensnummer (SYN+ACK). Til sist svarer klienten med en tredje TPDU aom bekrefter tjenerens sekvensnummer (ACK). Grunnen til at det ikke brukes forhåndsdefinerte initielle sekvensnummer er at TCP krever at hver side av forbindelsen selv velger disse tilveldig, for å hindre at to instanser av samme forbindelse gjenbruker de samme sekvensnummer for raskt.

7. *Fragmentering og reassemblering blir tatt hånd om av IP. Betyr det at TCP ikke trenger å bekymre seg om at data kan komme frem feil rekkefølge?*

   Det er kun de separate pakkene som blir satt tilbake til opprinnelig tilstand av IP etter fragmentering. Dvs at IP ikke besørger annet enn å ikke levere fragmenterte pakker fra IP og opp til transportlaget. Rekkefølgen av pakkene er likegyldig for IP, og TCP må håndtere situasjoner hvor pakker kommer frem i en annen rekkefølge enn de ble sendt i. Eksempelvis TCP sin sliding window algoritme tar seg av å stokke pakkene tilbake i rekkefølge.

2. Oppgaver basert på Computer Networks 4th edition

1. *Hva om vi kun hadde et to-veis håndtrykk. Hvorfor bruker vi ikke dette?*

   Tenk på dette scenarioet: Host 1 sender en SYNpakke til Host 2. Host 2 setter opp tilkoblingen sin og sender deretter en ACK tilbake på denne SYNpakken? Hva skjer om denne ACKpakken forsvinner på veien tilbake? I dette tilfellet har Host 2 allerede satt opp en tilkobling, mens Host 1 fortsatt venter på svar fra Host 2. En timeout vil forekomme hos Host 1 etter en stund og en ny SYNpakke vil sendes til Host 2. Problemet her er at Host 2 vil se på denne SYNpakken som en ny tilkobling mellom de to nodene, ikke som en resending av den første SYNpakken, og dermed heller sette opp 2 tilkoblinger til Host 1.

2. *Hva med et tre-veis håndtrykk? Løser dette problemene med to-veis håndtrykket?*

   Et tre-veis håndtrykk vil løse problemene som beskrevet over ved at begge partene i kommunikasjonen vil gå inn i en mellomliggende fase før den faktiske tilkoblingen settes opp. Hvis vi tenker oss samme scenario som i oppgave 1 så vil Host 2 sende en SYN/ACK pakke etter å ha mottatt den første SYNpakken fra Host 1. Etter at denne pakken er sendt vil Host 2 gå inn i en mellomliggende fase kalt SYN-SENT i stedet for å sette opp tilkoblingen direkte. Host 2 vil være i denne mellomliggende fasen helt til den mottar en ACK på SYN/ACK pakken som den sendte tilbake til Host 1. Hvis SYN/ACK pakken forsvinner i dette tilfellet, og Host 2 mottar en ny SYN pakke fra Host 1 så vil Host 2 forstå at SYN/ACK pakken må ha forsvunnet ettersom en ACK

aldri har kommet fram på denne pakken.

3. *Hva er "The two army's problem" og hvordan relaterer det seg til nedkobling av en connection i TCP?*

   Viser til side 537 i Computer Networks 5th edition (eller side 503 i 4th edition) for en forklaring av The two army's problem. Poenget her er at det vil være umulig å synkronisere de to armeene til blå. Det samme gjelder nedkobling av en connection i TCP. Det finnes ingen protokoller som garanterer at begge partene i en samtale tar ned tilkoblingen samtidig.

4. Trenger man egentlig UDP protokollen? Kunne man ikke like godt bare sende rene IP pakker?

   UDP implementerer ikke mye funksjonalitet, men protokollen har allikevel en viktig oppgave, nemlig å identifisere hvilken applikasjon pakken segmentet tilhører. IP protokollen har mulighet til å identifisere (og å finne) en viss maskin på Internett, men når denne pakken kommer fram trenger man en protokoll som sier noe om hvilken applikasjon pakken skal leveres til, og det er altså dette UDP gjør.

5. Både UDP og TCP bruker portnumre for å identifisere en prosess. Hvorfor oppfant man en ny type ID for prosesser (portnummer) i stedet for å bruke prosessIDen som allerede var i bruk da man designet disse protokollene?

   Det er tre grunner til at man bruker portnumre i stedet for prosessIDen som man finner i operativsystemet:
   1. En prosessID er spesifikk for hvert operativsystem, ved bruk av en slik prosessID måtte protokollen også vært skreddersydd for operativsystemet.
   2. I enkelte tilfeller vil man ha muligheten til å sette opp flere tilkoblinger mellom to prosesser, da må man ha en måte å skille disse på.
   3. Enkelte portnumre er såkalte "godt kjente" portnumre. Dette gjelder for eksempel HTTP som alltid bruker port nummer 80. Dette ville vært umulig å gjennomføre med prosessIDer.

6. Hvorfor er maksimum nyttelast (payload) for et TCP segment 65.495 bytes? 65536

   Grunnen til at et TCP segment kun kan være på 65.495 bytes er fordi et IP datagram max kan være på 65.515 bytes. Hvis man da har nyttelast på 65.495 bytes + en TCP header + 20 bytes er man på 65.515 bytes.

   Dette lar det være igjen 20 bytes med IP header som gir 65.535 bytes (som er resultatet av 2^16 - 1).

1. Metningskontroll (congestion control)

1. *Definer begrepene 'metning' og 'metningskontroll' i forhold til nettverk. Hvor og hvordan oppstår metning i et nettverk? Hva er hovedprinsippene for metningskontroll?*
   Metning er en tilstand i (en del av) et nettverk hvor det er så mange pakker på et gitt tidspunkt at det reduserer ytelsen. Kort sagt er ressursene overbelastet. Dette oppstår hovedsaklig i switcher/rutere når det samles (aggregeres) trafikk fra flere linker som skal

inn på en og samme link (pakker legges i samme utbuffer), eller når det er uoverensstemmelse mellom båndbredden på link inn og ut av en ruter langs pakkens path (linkspeed mismatch) slik at pakkene kommer inn raskere enn ruteren kan sende dem videre. Pakkene vil i begge tilfeller gradvis fylle opp tilgjengelig bufferkapasitet i ruteren, og i verste tilelle må overskuddspakker det ikke er plass til, kastes. *Metningskontroll* er mekanismer for å hindre/kontrollere metningen.

Hovedprinsippene for metningskontroll er 1) unngå situasjoner hvor metning oppstår ved gjennomtenkt nettverksdesign og kontrollert trafikkmønster. 2) lett påtrykket når metning er i ferd med å bygge seg opp: a) send info til noden som er opphavet til pakkene som skaper problem (the bad guy, oversvømmer nettet med pakker), b) send info til alle nabonoder og be dem alle ta det litt roligere (sprer metningen litt utover i nettet). Man kan iverksette tiltak som å øke kapasiteten til buffer og linker, endre forwardingstabeller for bedre balanse i trafikken, og/eller redusere senderaten. Det er uansett viktig å unngå at konrollen introduserer svingene oppførsel; bedre med en jevn strøm pakker enn situasjon som veksler mellom stille og oversvømmelse.

2. *Er det noen forskjell på metningskontroll i hhv. forbindelsesorienterte og -fire nettverk?*

   Ved oppstart av en forbindelse i forbindelesorienterte nett allokeres det ofte ressurser. Denne allokeringen vil unngå metning (forutsatt at man har anta riktig ang. forventet ressursbruk). På den annen siden er det da potensiale for å sløse med ressursene i situasjonene hvor forbindelsen ikke utnytter sin reserverte del maksimalt (eks venter på ack/retransmisjon). Det er imidlertid også en del forbindelsesorienterte nett hvor det ikke reserveres, men kun settes opp forbindelse, og i slike tilfeller vil man kunne se metning. I forbindelsesløse pakkesvitsjede nettverk (Internett) preallokeres det ikke noen ressurser, og man må forvente og håndtere situasjoner med metning i ruterne.

3. Forklar begrepene *back pressure* og *soft state*

   Back pressure: man bruker en mekanisme som holder tilbake pakker i buffre i oppstrøms noder, sett fra den ruteren som har metningsproblemer. Man skyver altså problemer bort fra og bakover i nettverket, og bruker selve nettet (og dets kollektive bufferressurser) til å bufre pakkene istedenfor i kun èn node. Alternativet er å kaste pakker hvis man ikke klarer å redusere belastningen for den aktuelle ruteren.

   Soft state: Ressursallokeringen i ruterne er dynamisk, basert på informasjon om dataflyter. I et forbindelsesfritt pakkesvitsjet nettverk blir alle pakker i prinsippet svitsjet uavhengig av hverandre, men i praksis utveksler to kommuniserende noder en strøm av meldinger. Tilstandsinformasjonen endres altså over tid, og gjør at man kan ta mer velbegrunnede avgjørelser, i motsetning til ved 'hard state' hvor info er hardkodet på forhånd.

4. *Redegjør for hvilke lag i OSI-modellen vi finner metningskontroll, hvordan mekanismen fungerer på disse lagene, og evt hvordan de samspiller (utnytter tjenester).*

   Datalinklaget: Retransmisjon, kvittering, (hop-by-hop) flytkontroll, caching av out-of-order pakker. *Nettverkslaget*: Rutingalgoritmer, kømekanismer, betjeningsrekkefølge, kastepolitikk, håndtering av pakkelivssyklus(TTL etc).*Transportlaget*: Retransmisjon, kvittering, (ende-til-ende) flytkontroll, timeoutverdier, cacing av out-of-order pakker.

   Linklaget og transportlaget håndterer omtrent de samme elementene av metningskontrollen, forskjellen ligger i at linklaget styrer i utgangspunktet den enkelte link, og er best egnet til å ta seg av kortvarige metningssituasjoner og kontroll av bufferutnyttelse, mens transportlaget håndterer metning på et ende-til-ende nivå og dermed kan ta seg av mer langvarige metningssituasjoner og "permanent" redusere

senderaten til aggressive kilder. Nettverkslaget håndterer veivalg og selve skeduleringen i den enkelte ruter, og er på en måte en mer lokal del av metningskontrollen enn lagene over og under (selv om topologi og overordnet rutingtabell selvfølgelig også angår samtlige noder).

5. *Redegjør for ulike typer kødisiplin (relatert til switcher/rutere) , og diskuter fordeler/ulemper ved disse typene.*

   FIFO: First in first out .. kun èn kø, hvor det ikke tas hensyn til hvilken flyt en pakke tilhører. Vanlig å implementere fifo med taildrop, dvs at pakker kastes fortløpende når køen er full. RED og DECbit algoritmene er imidlertid merkompliserte når det gjelder valg av hvilken pakke som skal kastes.
   FQ: Fair queuing .. De ulike flyter plasseres i ulike køer som behandles round robin. Hvis det med denne kømekanismen sendes data for raskt i en flyt, vil problemet være isolert til den ene køen, og ikke ramme de andre flytene. Det er et designspørsmål hvor mange slike del-køer man skal tillate, og det er mulig at flere flyter må dele en og samme kø, feks hvis en sorterer flyter etter destinasjonsadresse. En variant av FQ (weighted fq) tillater at del-køene har ulik vekt basert på prioriteten til de data som ligger i køen

6. *Beskriv ulike strategier for å unngå metning.*

   DECbit: Ruteren gir tilbakemelding til noder før metning intreffer ved at det settes et bit i pakkene som passerer ruteren. Mottaker kopierer så dette bitet inn i ACK slik at avsender tilslutt får beskjed om at metning er under oppbygging. Kriterier for å sette bitet er ofte at den gjennomsnitlige kølengden er større en 1 på det tidspunktet pakken passerer ruteren.

   RED: Random Early Detection (brukt sammen med TCP) gir tilbakemelding etter samme prinsipp som over, men istedenfor å sette et bit (eksplisiv feedback), gis det kun implisitt feedback i form av pakketap. Dvs at det kastes en pakke i ruteren før det strengt talt er behov for det. TCP avsenderen vil detektere pakketapet ved uteblitt ack på aktuelt sekvensnummer og justere senderaten sin. Fordelen er at det advarende pakkekastet medfører at en kan iverksette tiltak tidligere og om mulig unngå massivt pakketap som er alternative når man venter på full metning før noe skjer.

   Alternative mekanismer overvåker RTTverdier, og tolker økning i gjennomsnittlig RTT som indikasjon på at en eller flere rutere er i ferd med å bli overbelastet. Et alternativ er å sammenlikne nåværende RTT med snittet av min og max RTT, og redusere metningsvindu med 1/8 hvis større. Alternatvt kan en sammenlikne throughput for hver RTT.

7. *Forklar hvordan metningskontroll er implementert i TCP, og knytt din forklaring til begrepene* .

   Implementert som kontroll av avsendervinduet(congestion window) ved hjelp av additiv increase / multiplicative decrease, slow start, fast retransmission, fast recovery osv. *Slow start* brukes for å øke metningsvinduet raskt fra en "kald start". Økningen er eksponensiell istedenfor lineær, slik at vinduet fordobles per RTT inntil pakketap, hvorpå det halveres og man går over å fortsette med additiv increase. Denne mekanismen benyttes både etter oppkobling av en forbindelse, og etter at at en metning har opphørt. *Additiv increase* vil si at for hver burst(fullt vindu med pakker i løpet av en RTT) økes vinduet med 1, imotsetning til slowstart hvor hver enkeltpakke medfører denne økningen.

   Fast retransmit vil si å resende pakker med det samme det oppdages at de er borte, fremfor å vente på ack en hel RTT først. Mottaker vil sende ack på pakke n-1 (sist mottatte pakke i sekvens) på hver mottatte pakke n+1 helt til n mottas. Det er altså det høyeste pakkenummeret som til envher tid har ankommet korrekt som settes i ack,

imotsetning til det å ack'e med sekvensnummer tilsvarende pakken som kom. Gjentagende "gamle" ack-verdier vil da signalisere til avsender at pakken er savnet/borte.

Fast recovery: iverksettes ved metning ved at metningsvinduet halveres etterfulgt av additiv increase.

8. Anta at vi har en rundetid på 10 msec og ingen metning. Mottakervinduet er på 24 KB og maksimumssegmentet (maximum segment size/MSS) er 2 KB. Hvor lang tid tar det før det første fulle vinduet blir sendt?

   Første runde sendes 2 KB (MSS), deretter 4KB, 8KB, 16KB og til slutt 24 KB (avsenderen sender aldri mer enn mottakervindet). En rundetid på 10ms og 4 runder for å komme opp i 24 KB skulle gi 40 ms.

9. Anta at TCP sitt metningsvindu er på 18 KB og en timeout forekommer. Hvor stort vil vinduet være hvis de neste 4 tranmisjonene er vellykkede?

   Når en timeout forekommer vil TCP anta at nettet er mettet og vil dermed gå inn i en ny slow start fase. Dette vil si at første transmisjon etter en timeout vil være på størrelse med MSS (maximum segment size). Hvis vi antar at MSS er 2KB vil vi første runde sende 2KB, deretter 4KB, 8KB og etter 4 transmisjoner 16KB.

10. En TCP maskin sender fulle vinduer på 65.535 bytes over en 1-Gbps linje med en 10 msec forsinkelse. Hva er maksimal gjennomstrømning (throughput)? Hvor mye av linjen utnyttes?

    Her tar det 10ms å sende et fullt vindu over til mottakeren, og deretter 10ms for å vente på ACK. Dvs. at vi kan sende ett vindu per 20ms. Da rekker vi 50 vinduer på 1 sekund. 50 vinduer * 65.535 bytes = 3,3 millioner bytes.
    3,3 millioner bytes regnet om til Mbit skulle gi 26,4 Mbit. Så med en vindusstørrelse på 65.535 bytes kan vi maksimalt sende 26,4 Mbit/sec data. Vi kan tydelig se at vindusstørrelsen begrenser hvor mye data vi kan sende. 26,4/1000 * 100 gir oss svaret som sier at vi kun utnytter 2,6% av linja.

2. RPC

1. *Hva er RPC, og beskriv hvordan det fungerer.*

Remote Procedure Call er en teknikk som muliggjør det å kalle en prosedyre i et annet adresserom (feks. i en annen maskin). Hensiktet er at RPC skal minne mest mulig om vanlige prosedyrekall, dvs med parametre, returverdi og mulige exceptions, og det skal være så enkelt som mulig for programmereren. Mest mulig av kommunikasjonsdetaljene skal skjules (transperent). Kall som skal være tilgjengelige som RPC, må spesifiseres vha. et *interface definition language*(IDL) som er et deklarativt språk bestående av grensesnitt til prosedyrene. IDL inneholder ikke implementasjonsdetaljer.

En IDL kompilator vil lage *stubs* for klient og tjener. Når en klient kaller en fjernprosedyre vil kallet gå til klientstubben, hvor kallet endres til et passende nettverksformat. En egnet protokoll (eks TCP/IP) brukes til å overføre kallet til tjenerstubben, hvor meldingen konverteres tilbake til et lokalt prosedyrekall som eksekveres på tjeneren. Selv om all kommunikasjon går via stubbene, er dette skjult for programmerer.

Stegene for å lage en RPCapplikasjon er som følger:

1. Lag IDL'en. Definisjoner i denne er tilgjengelig for klienter.
2. Kompiler IDL'en. Produserer headerfiler samt stubs for klient og tjener.
3. Programmer tjenerens kode. De prosedyrene som spesifiseres i IDL må nå implementeres. Kompileres sammen med headerfiler og tjenerstub.
4. Programmer klientens kode. Her kan man invokere fjernprosedyrekallene. Kompileres sammen med headerfiler samt klientstub.

se man rpc for mer info om hvordan rpc benyttes sammen med C.

2. *Redegjør for de fire ulike kallsemantikkene RPC kan operere med .*

En RPCimplementasjon kan ha ulike leveringsgarantier, også kalt RPCsemantikk. Hvilken semantikk-form RPCimplementasjonen støtter er svært viktig å vite. Det som skiller semantikkene fra hverandre er 1) om man forsøker retransmittere meldinger til det kommer svar, eller antar at tjeneren er død. 2) om man filtrerer ut duplikate meldinger ved retransmisjon 3) om man holder oversikt over svarene tjeneren har sendt, slik at disse kan retransmitteres ved behovuten å måtte utføre det originale prosedyrekallet på nytt.

exactly-once:et prosedyrekall fra klient til tjener ugføres nøyaktig 1 gang, noe som er vanskelig, hvis ikke umulig, å garantere for. Problemet er at en klient ikke ser forskjell på en feil i nettverket (pakkekast ved metning) og feil i tjeneren (core dumped). Man kan garantere denne semantikken hvis tjeneren ikke kræsjer og klienten mottar resultatet av kallet.

at-most-once: er den vanligste semantikken, og innebærer at kallet fra klient til tjener utføres eller ikke(0 eller 1 gang). Tjeneren filterer ut duplikate meldinger fra klienten.

at-least-once: vil si at kallet fra klient til tjener utføres en eller flere ganger. Meldinger retransmitteres, uten at tjeneren filtrerer duplikater.

maybe: Prosedyrekallet fra klient til tjener utføres kanskje, dvs at klienten ved timeout ikke retransmitterer meldinger. Man kan ikke si med sikkerhet om kallet ble utført: hvis meldingen ble borte eller at tjeneren kræsjet, ble ikke kallet utført. Imidlertid hvis det var svarmeldingen som ble borte, så kan kallet ha forekommet.

2. Networklayer and Routing

Referanse: Kap 5

2.1. Prinsipper og begreper

Oppgave 27:

Hva er de 3 viktigste hovedoppgavene til Nettverkslaget?

Nettverkslaget tilbyr kommunikasjon mellom hosts (verter/maskiner), i motsetning til Transportlaget som tilbyr kommunikasjon mellom prosesser.

De tre viktigsste hovedoppgavene til nettverkslaget er: Path determination (beregne rute), ruting (transportere datapakker til riktig node i henhold til beregnet rute) og call setup (oppsett av forbindelse)

Call setup – ATM opplegg.. forbindelse settes opp på forhånd.. i virkeligheten kjører vi IP over ATM.. ikke som i TCP der vi har three way handshake

Oppgave 28:

En *service model* for nettverkslaget må si noe om støtte for *garanteed bandwidth*, *preservation of inter-packet timing (no jitter)*, *loss-free delivery*, *in-order delivery* og *congestion feedback to sender*. I den sammenheng snakker vi for nettverkslaget om to konseptuelle service modeller. Hvilke?

Datagram og Virtual Circuit

Datagram: sende pakker.. ingen garantier for noen av de ovennevnte ting..

ATM er eksempel på virtual circuit..

Virtual Circuit kan tilby ovennevnte..

Hvordan får vi pakkene våre gjennom et nettverk?
(circuit switching, packet switching)

Circuit switching (krets switching kan deles opp i FDM (Frequenzy) og TDM(time)): dedikert linje per kall.. ala gammeldags telefonlinje.. eller en direkte linje mellom to punkt.. i dag er det i praksis ikke slik, men prinsippet er det samme..
→tenk på i gamle dager når du dessa sekretærane flytta fysisk en kabel over på riktig punkt i en sentral..
Packet switching: data blir sendt gjennom et nett i små "biter" eller pakker…
→Skiller på datagram-nettverk og Virtual Circuit (VC)
Eksempel på VC: ATM, Frame Relay, X.25

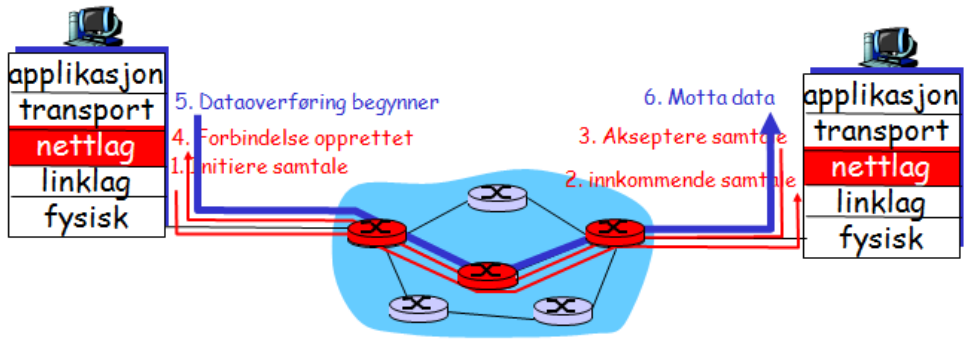
Oppgave 29:

Beskriv hovedtrekkene ved en Virtual Circuit (VC) service model. Bruk gjerne eksempler fra ATM?

VC setup (call setup): *Transportlaget til avsenderen kontakter nettverkslaget. Når dette blir gjort sendes adressen til mottakeren med slik at nettverkslaget kan beregne/bestemme hvilke linjer og hvilke rutere pakkene skal gå via for å komme frem til mottaker. Alle pakkene vil bli sendt denne veien. Nettverkslaget bestemmer også VC-id for hvert hopp (fra ruter til ruter). Til slutt blir det lagt til en post i rutingtabellen til hver ruter som pakkene skal gå via. Det er nettverkslaget som gjør dette. Nettverkslaget kan også reservere ressurser som båndbredde under denne prosessen.*

Dataoverføring: *Når VC er satt opp kan data sendes med en gang.*

VC nedkobling (teardown): *Dette skjer når sender eller mottaker gir beskjed til nettverkslaget om at den vil avslutte tilkoblingen (VCen). Nettverkslaget informerer så enheten i den andre enden om dette. Videre oppdateres rutingtabellene for å angi at VCen ikke lenger eksisterer.*



Oppgave 30:

Beskriv hovedtrekkene ved en Datagram service model?

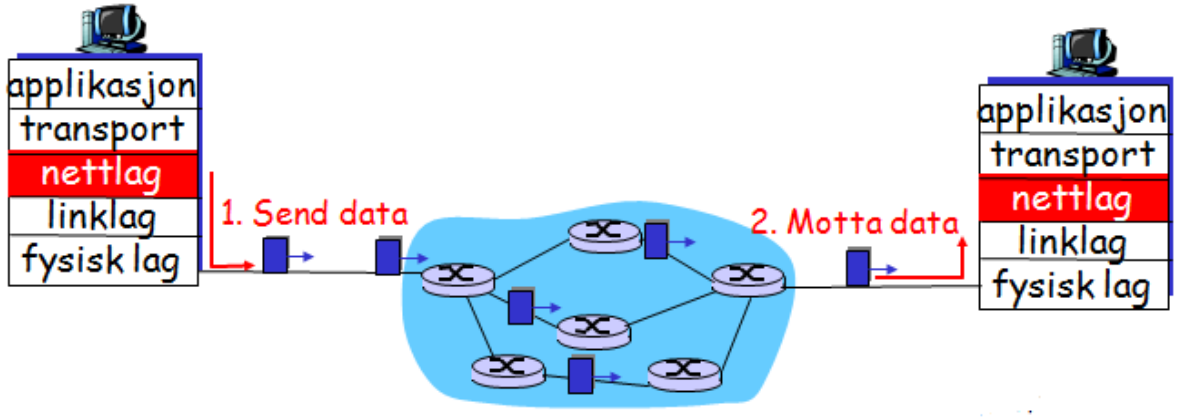
En Datagram service modell er noe helt annet enn VC. I motsetning til VC som oppretter forbindelse mellom avsender og mottaker, blir det her ikke opprettet noen form for forbindelse før sending av data. Dette gjør at ikke alle pakkene trenger å ta samme vei gjennom nettverket. Pakkene videresendes fra ruter til ruter ved hjelp av ipadressen. Ruterne beregner neste hopp ved hjelp av rutingtabeller. Måten den finner ut hvor den skal sende pakken videre er at den bruker prefix match. Dette betyr at den matcher ipadressen med det som ligger i rutingtabellen. Dersom det er flere som matcher ett stykke på vei, videresender den pakken på den porten som har best match, også kjent som "longest prefix matcing rule". Dersom ingen matcher vil den benytte seg av default gateway, også kjent som "otherwise".

Oppgave 31:

Hva er *ruting* og hva er det overordnede målet med *ruting*?

Mange tror at ruting er å sende pakken videre fra et nettverk til et annet. Dette er på en måte rett, men dette betegnes som forwarding. Ruting derimot er det som gjøres før forwardingen, altså å planlegge hvilken port pakken skal sendes videre på.

Disse to begrepene kan sammenlignes med en kjøretur fra Ålesund til Oslo. Ruting vil her være å planlegge hvor man skal kjøre for å komme til Oslo, kanskje ved hjelp av kart. Forwarding vil da være å kjøre fra Ålesund til Oslo.



Hva menes med begrepet *cost* ifm. Routing (Ruting) ?

Cost er et måltall som brukes for å si noe om hvilken rute gjennom et nettverk som bør velges for å best tilfredsstille avsender og mottaker. Vanligvis forbindes Cost med hvor stor forsinkelse, eller hvor liten båndbredde som er tilgjengelig via en valgt rute, men cost kan også være andre faktorer også. Ved ruting, prøver man altså å finne den ruten gjennom nettet som gir laves cost-tall.

Oppgave 32:

For å beregne rute gjennom netteverket, benyttes ruting-algoritmer. Hvordan klassifiserer vi ruting algoritmer?

Global eller desentralisert?

Ved global ruting-algoritme har alle ruterne fullstendig info om topologi og cost på de andre ruterne på ruten. Dette kalles en "link-state"-algoritme.

Ved desentralisert ruting-algoritme kjenner ruterne bare info om de andre ruterne som er direkte tilknyttet seg selv. Her må ruterne utveksle info med hverandre for å finne ut topologi og cost videre på ruten. Dette kalles en "distance vector"-algoritme.

Statisk eller dynamisk?

En annen måte å klassifisere ruting-algoritmer på er om de er statiske eller dynamiske. Statisk betyr at ***rutingtabellen?????*** *er satt opp manuelt/statisk. Denne endres sjelden, og dette gjøres som regel manuelt. En dynamisk ruting-algoritme oppdateres hele tiden. Denne kan stilles inn til å oppdateres for eksempel en gang i minuttet eller den kan oppdateres automatisk når topologi eller cost forandres. Dynamisk ruting-algoritme reagerer lettere på netverksforandringer, men den er også lettere mottakelig for feil som kan oppstå, for eksempel ruting-looper.*

Load-sensitive eller load-insensitive?

En tredje måte å klassifisere ruting-algoritmer på er load-sensitiv eller load-insensitiv. I en load-sensitiv algoritme, endres link-cost dynamisk ut ifra overbelastning på den underliggende linken. Dersom høg kost knyttes til en link som er overbelastet vil routing algoritmen prøve å finne en rute rundt denne linken som har lavere kost.

Load insensitive?????

Oppgave 33:

Beskriv prinsippene ved en *Link State* algoritme og en *Distance Vector* algoritme. Vektlegg ulikhetene.

Link State: Alle noder i nettet kjenner alle andre noder samt linker mellom nodene i hele nettet. Rutingtabell i en gitt node beregnes med minste kosts rute til samtlige av de andre nodene i nettet.

Distance Vector: En gitt node kjenner kun sine naboer og linkene til sine naboer. Rutingtabellen til en gitt node vil inneholde kost til en eller flere destinasjoner, samt hvilken nabonode en rute må gå via for å nå destinasjons noden.

Oppgave 34:

Hva er fordelene og ulempene med henholdsvis *Link State*- og *Distance Vector* algoritmene?

Link-state: Fordel: Rask algortime ved få noder. Endringer i cost blir raskt oppfattet. Ulemper: Alle noder må kjenne alle andre noder og linker i nettet. Lite skalerbar, lite egnet i store nettverk. Med mange noder: treg algortime.

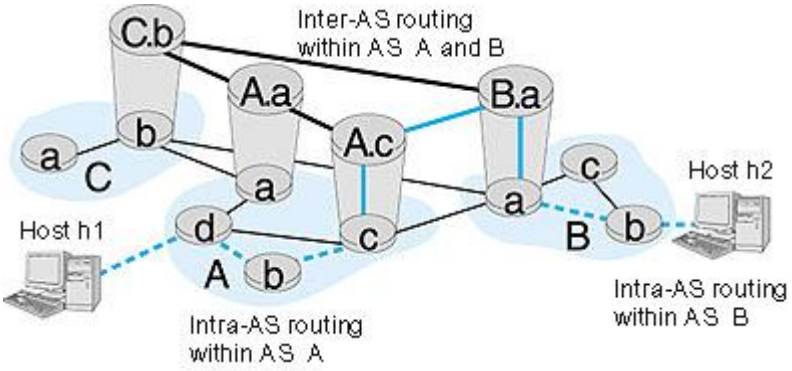
Distance-vector: Fordeler: Skalerbar. Velegnet til større nettverk. "Good-news-travels-fast". Ulemper: "Bad-news-travels slow". Routing loops.

Oppgave 35:

Hva menes med *hirarkisk ruting (Hierarchical Routing)*. Forklar I den sammenheng hva som menes med et *Autonomt System (Autonomous system) (AS)*?

Ved hierarkisk ruting grupperes rutere i autonome systemer, heretter betegnet som AS. Et AS er en gruppe med rutere som har felles administrator, for eksempel innenfor en ISP eller innenfor ett bedriftsnettverk. Alle rutere innenfor et AS kjører samme ruringalgoritme. Rutingalgoritmen som kjøres innenfor et AS kalles for "intra-as routing protocol". Inter-as routing protocol er algoritmen som kjøres mellom ulike AS. For at rutere på Internett skal kommunisere med hverandre må alle disse kjøre samme inter-as ruting protokoll. Den som er i bruk i dag kalles BGP4 (Border Gateway Protocol 4). BGP brukes for å gi beskjed til resten av Internett om at et subnett eksister, og hvor det befinner seg. Om vi ikke hadde hatt BGP ville alle subnet vært isolert og utilgjengelig for resten av Internett.

Oppgave 36:



Figuren over viser eksempel på hirarkisk ruting. Hva kalles en ruter som både skal forholde seg til Inter-nettverket og Intra-nettverket (som f.eks. ruteren C.b)?

En slik ruter kalles en Gateway-ruter.

Beskriv hva som skjer når Host h1 skal kommunisere med Host h2.

Host h1 kontakter ruteren A.d som er satt opp som default gateway i ip-innstillingene på maskinen (Dette må ikke forveksles med en Gateway-ruter). Deretter sendes pakken videre til ruter A.b som igjen sender pakken videre til A.c. Nå har pakken nådd enden på AS A, og router A.c, som er en av gateway-ruterne til AS A, sender pakken videre til router B.a som er gateway-ruteren til AS B. Videre blir pakken sendt til router B.b gjennom B.c

2.2. Nettverkslaget på Internett, IP

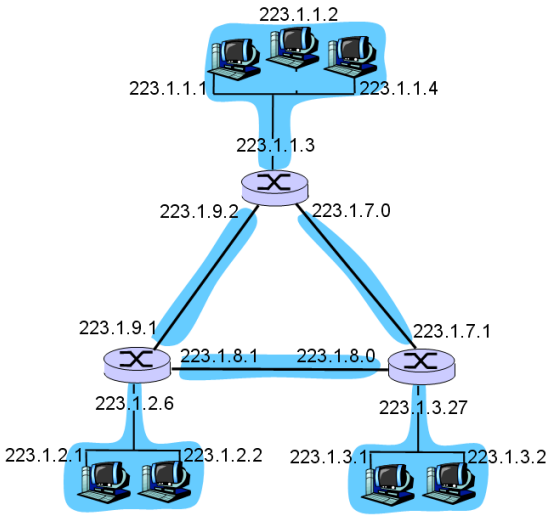
2.2.1. IP og nettverk

Oppgave 37:

Hvilke tre hovedkomponenter består Internet Nettverks laget av?

Ruting protokoller *(RIP, OSPF, BGP etc.),* ***IP protokollen*** *(selve databæreren) og* ***ICMP protokollen*** *(feilrapportering, ruter signallering etc.).*

Oppgave 38:



Hva er IP-adressen egentlig adressen til? (maskinen, nettverksinterfacet eller noe helt annet?)

IP-adressen er adressen til et nettverksinterface siden en maskin kan ha flere nettverksinterfacer. Et nettverksinterface har også en MAC-adresse, men denne befinner seg på linklaget i OSImodellen og ikke på nettverkslaget.

Med bakgrunn i figuren over, forklar hva som i IP-sammenheng menes med et nettverk?

Et nettverk i IP-sammenheng er en begrenset IP-serie eller et subnet hvor maskiner nettverksenheter kan kommunisere med hverandre uten å gå via en ruter. Et nett kan ikke være mindre enn 2^2, det vil si 4 adresser totalt. Den første adressen i ett nettverk kalles Subnet ID og den siste kalles Broadcast adresse. Dette medfører at om man har et nettverk på 2^2 har man allerede brukt opp 2 av adressene til subnet id og broadcast adresse. Så da står man igjen med 2 adresser som kan brukes, en til for eksempel en ruter og en til en pc. Man kan selvfølgelig unnlate å ha en ruter og heller ha to datamaskiner, men disse vil da ikke kunne kommunisere med andre nettverk, for eksempel internett.

Hvor mange IP-netverk har vi i figuren over?

I figuren er det 6 IP-nettverk. Disse er:

- *233.1.1.* - Ett nettverk med datamaskiner (øverst på bildet)*
- *233.1.9.* - Nettverket mellom ruterne som har 1.1 og 1.2*
- *233.1.7.* - Nettverket mellom ruterne som har 1.1 og 1.3*
- *233.1.8.* - Nettverker mellom ruterne som har 1.2 og 1.3*
- *233.1.2.* - Ett nettverk med datamaskiner (nede til venstre)*
- *233.1.3.* - Ett nettverk med datamaskiner (nede til høgre)*

Oppgave 39:

Hva mener vi med NAT, og i hvilken sammenheng brukes dette (gi gjerne eksempler)? Dette gir økt sikkerhet for et nettverk. Hvorfor?

NAT(Network address Translation) ideen er at ISP skal gi hvert hjem eller firma sitt eget IP addresse for internettrafikken. Innenfor kundens nettverk, får hver pc sitt unike ip addresse for brukt for ruting i trafikken. Men før pakket sendes og går gjennom ISP, må addressen oversettes fra ip addresse til det delte ip addressen. Dette gir økt sikkerhet fordi NAT inneholder brannmur og er forsiktig med hva som kommer inn og ut til kundens nettverk.

Rutingprotokoller på Internett

Generellt

Oppgave 40:

Hvilke ruting-protokoll(er) benyttes på Internett for henholdsvis Intra-AS og Inter-AS ruting ?

Intra AS: RIP, OSPF, IGRP (både link state og distance vector)

Inter AS: Path vektor(utvidelse av distance vector)

RIP

Oppgave 41:

Beskriv kort RIP?

RIP(Routing Information Protocol) er en type distansevektor hvor det utveksles hvert 30 sek via en respons message(advertisement) og hvert advertisement inneholder opp til 25 destinasjonsnett. Dersom intet advertisement hørt etter 180 sek, antar man at nabo eller link er død, alle siter via naboen blir invalidert, nye advertisement blir sendt til de andre naboene, naboene sender da ut advertisements dersom deres tabeller har endret seg. Infromasjon om feilende linker propagerer hurtig og poison reverse brukes til å hindre ping pong loops(uendelig distanse).

RIP blir vedlikeholdt av en applikasjonsprosess kalt route-d advertiesements innkapsles i UDP pakker(pålitelig levering ikke krevd, advertisements repeteres jevnlig).

OSPF

Oppgave 42:

Beskriv kort OSPF?

Bruker LINK STATE OSPF advertisement inneholder en entry per naborute. Advertisements dissemineres til hele Autonomous system. OSPF sine egenskaper er sikkerhet hvor alle OSPF meldinger blir autentisert, TCP forbindelse blir brukt. Den har multiple stier med samme kostnad tillatt for hver link, multiple kost metrikker fro forskjellige anvendelser.

Den har integrert multi og unicast støttet.

Den er satt hierarkisk OSPF i store domener.

Hierarkisk OSP er satt opp i to nivåer: local area og backbone.

Link state advertisements forlater ikke sitt domene. Noder i hvert domene har detalsjert domenetopologi: de kjenner bare retninger (shortest path) til noder i andre domener.

Area border rutere: sammenfatter avstander til nettverk i domenet, og informerer andre Adrea Border routers.

Backbone rutere: kjører en OSPF ruting alg begrenset til backbone.

Boundary rutere: kobler seg til andre AS'er.

Ruter Arkitektur

Oppgave 43:

Beskriv de 2 viktigste funksjonene til en Ruter?

Bestemme hvilke vei pakken skal bli sent(forwarding algorithm) .

Oppgave 51

Hvorfor trenger vi en ny versjon av IP-protokollen?

Vi trenger en ny versjon av IP-protokollen fordi IPv4 protokollen vil gå tom for adresser (32 bit), IPv6 bruker 128bit for adressering så vi får veldig mange nye adresser.

Beskriv de viktigeste endringene i IPv6 i forhold til IPv4?

Den viktigeste endringen på den nye protokollen er 128 bits for adresser istedenfor 32, den nye protokollen har også endel andre endringer i header, flere felt blir tatt bort siden man så at man egentlig ikke hadde bruk for de. Sjekksummen i IP headeren har blitt tatt bort, noe som øker hastigheten en router kan håndtere en IPv6 pakke på.
IPv6 kan sende "jumbopakker" på optill hele 4GB (dersom linker støtter dette, backbone nettverk o.l.).

1. Nettlaget - generelt

- *a) Gi en beskrivelse av hovedoppgavene til nettverkslaget, og knytt dette opp til begrepene pålitelig/upålitelig og forbindelsesfri/forbindelsesorientert.*

  Hovedoppgaven til nettlaget er å transportere pakker fra SAP på toppen av avsenders nettlag, gjennom nettet og til SAP på toppen av mottakers nettlag. Ofte vil pakkene måtte passere mange rutere underveis. Nettlaget er det laveste laget som tar seg av ende til ende transmisjon, i motsetning til linklaget som håndterer flytting av rammer over enkeltlinker. Nettverkstjenesten kan være forbindelsesorientert, eller -fri. Uavhengig av dette er oppgavene til nettlaget adressering og ruting. Dersom tjenesten er forbindelsesorientert, må nettlaget i tillegg ha støtte for glidende vindu, flytkontroll, og metningskontroll. Upålitelig forbindelsesfri tjeneste representert ved Internett, imotsetning til forbindelsesorientert pålitelig tjeneste som kan knyttes til telefonsystemer.

- *b) Gi eksempler på lag 3 protokoller i TCP/IP-modellen som er hhv forbindelsesorienterte og -løse.*

  TCP og UDP hhv

2. Nettlaget - Ruting/forwarding

- *Forklar hva som liggger i de tre datagram, virtuell forbindelse og kilderuting. Beskriv fordeler og ulemper ved dem.*

  Datagram: hver pakke inneholder den komplette adressen til destinasjonen. Switchen/ruteren benytter så denne informasjonen til å avgjøre hvor pakken skal sendes videre. Datagram er veldig enkle å sende, og kan sendes til enhver tid til enhver node, uten at avsender trenger bekymre seg for hvilken vei pakken tar. Imidlertid er det ikke garantert at datagram når målet sitt; de kan gå tapt på veien, og avsender har ingen måte å finne ut av dette. En annen effekt, som normalt er uønsket, er at den innbyrdes rekkefølgen av datagram kan endres, noe som kan forekomme når ulike datagram følger ulike veier gjennom nettet. Denne rutingmetoden medfører også noe kompleksitet for ruterne når det gjelder å bygge rutingtabeller. En siste draw-back er at samtlige pakker må inneholde destinasjonsadressen.

  Virtuell forbindelse (virtual circuit, VC): en VC vil si at alle pakker fra en bestemt node, X, følger samme vei til en bestemt node, Y. For at en VC skal fungere må det først opprettes en forbindelse, ved at den host som ønsker å opprette forbindelsen sender en setup-message. Denne spesielle pakken vil da traversere nettet og sette av ressurser i de ruterne den er innom. Hver ruter som kan tilby VC'er, må vedlikeholde en tabell over de VC'er den har. Fordelen er at hver pakke som sendes kun trenger inneholde VCI, noe som er betydelig mindre enn en komplett adresse. I tillegg besørger VC at alle pakkerekkefølgen bevares. På den annen side er VC'er sårbare overfor brudd i kretsen; skulle en link/ruter gå ned må hele kretsen reetableres.

  Kilderuting: avsenderen er ansvarlig for at pakken inneholder den info som behøves for å få den frem til mottaker. Dvs at den må legge inn samtlige adresser/porter pakken skal passere underveis. Fordelen her er at ruterne blir svært enkle, mens avsenderen blir mer kompleks ved at den må kjenne til hele nettverket. Overhead i pakkene kan bli uforskammet stort, avhengig av hvor mange rutere pakken skal passere.

- *Hvilke hovedklasser rutingalgoritmer finnes for nettverkslaget? Gi eksempler, og forklar kort hvordan de virker.*

  ikke-adaptiv ruting: rutingtabellene beregnes på forhånd basert på topologi og evt. antatt belastning.

adaptiv ruting: rutingtabellen kan **ikke** forandres dynamisk når linjer brytes eller trafikk endres.

De tre hovedklassene algoritmer er distance vector, link state og hierarkisk ruting. *Distance Vector* er ruting med avstandsvektorer (Bellmann-Ford algoritmen, ARPANET inntil 1979). Hver ruter opparbeider en tabell med avstand og vektor til alle andre noder. Nodene kjenner i utgangspunktet bare sine egne naboer, men ettersom denne info spres i nettet , kjenner alle nodene avstand og veil til alle andre. Dette gjøres ved at alle noder informerer sine naboer om sin egen kjennskap til andre noder i form av (Destination,cost) tupler. Nodene som mottar disse vil da oppdatere sine interne rutetabeller bestående av (Destination,Cost,Nexthop)-tupler, og bringe disse videre til sine egne naboer. Denne info spres i to tilfeller: ved faste protokollavhengige intervall, samt ved såkalte trigged updates (når man har oppdatert sin egen tabel)

Link state ruting er basert på Dijkstras shortest path algoritme hvor alle noder er kjent med hele nettets topologi. I Internett benyttes Open Shortest Path First. Flodbølgeprinsippet benyttes for å etablere graf over topologi. Linktilstanden spres ved hjelp av såkalte link-state packets (LSP), som inneholder følgende: ID til LSP sitt opphav, liste av direkte tilkoblede naboer og kost tilknyttet disse, sekvensnr og TTL.

- *Datagram må rutes som separate pakker, mens alle pakker i en forbindelsesorientert transmisjon følger samme vei. Betyr dette at nett som støtter virtuelle kretser aldri trenger å rute isolerte pakker mellom to steder?*

  For å rute en enkelt pakke mellom to steder i en forbindelsesorientert omgivelse, må man likevel etablere og koble ned en VC. Man må derfor koble opp, sende pakke og koble ned med all den overhead dette innebærer. Det er selvsagt ikke umulig å sende enkeltpakker over en VC, snarere et spesialtilfelle.

- *Vil pakker som sendes gjennom en virtuell krets noen gang komme frem i en annen rekkefølge enn de ble utsendt i ?*

  Dette vil ikke være ønskelig, fordi man i så fall må innføre buffring i endemaskinen og sortere pakkene ved mottak. Man kunne altså like gjerne ha benyttet datagram. Reordning kan heller ikke forekomme siden samtlige pakker passerer de samme rutere i samme rekkefølge

1. Ruting grunnlager

1. Hva er gode egenskaper for en ruting-algoritme?
   - korrekthet - få pakker fram til mottakeren
   - enkelthet - lett å implementere
   - robusthet - overleve mange feilende maskiner og linker, og klare å håndtere store endringer i topology og i vurdering av linkkvaliteter.
   - stabilitet - ikke noen store endring i rutingtabelle eller rutene som pakker ta etter en liten endring i linker som er tilstede eller i metrikken som vurderer linker.
   - fairness - kostnaderne for å overføre pakker mellom enhver par av sender og mottaker skal være omtrent like billig.
   - optimalitet - rute slikt at den samlete kostnaden for pakkeoverføring er minimalt. Kostnad i denne samheng er metrikken for ruter, f.eks. hvis kvalitetsmetrikken for ruter er antall hops, så er rutingalgoritmen optimalt som klarer å få pakker i gjennomsnittet fram over færrest mulig hops.

2. Gi et eksempel for motstridende gode egenskaper i en ruting-algoritme.
   - Fairness og optimalitet er de beste kandidatene for problemer, fordi den ene minimerer gjennomsnittet, den andre minimerer varians i kostnader for å overføre en pakke fra sendere til mottakere.
3. Hva er optimalitetsprinsippet i ruting?
   - Hvis ruter J ligger på den optimale veien V fra ruter I til ruter K, så er den optimale veien W fra J til K en del av V.
   - Og alle logiske variasjonene av dette.
4. Hva er et sink-tre, og hva er sammenheng mellom sink treet og optimalitetsprinsippet?
   - Gitt en ruter J i et nettverk, så vil sette av de optimale veiene fra alle andre rutere i nettverk være et tre. Hvis det ikke var tilfelle, men man kunne bruke en mer generisk graf, så hadde man en node I som ligger på de optimale veiene fra ruterene K og H til node J, men med forskjellige optimale delveier I-J for veien K-J og for veien H-J. Optimalitetsprinsippet sier at dette bare er mulig hvis begger veier I-J har samme kostnad. Men da kunne man også bruke av de to veiene I-J i begge tilfeller. Gjør man det for alle slike tilfeller, oppnår man en trestruktur.
5. Hva er grunnen for at sink-treer ikke bygges for å lage optimale ruting-tabeller i ekte nettverk?
   - Optimaliteten gjelder bare til et vist tidspunkt eller i et nettverk uten annen traffik.
   - Hvis kvalitetsmåling av linker med utvikling over tid inngår i algoritmen, så tar det bare for lang tid å utregne sink-treer.
   - I statiske nettverk (som sentral ruting med full statisk kunnskap over hele nettet og) kan man faktisk bruke sink treer på grunnlag av informasjon om det statisk nettverket uten noen last. Men også da ville optimaliteten være problematisk fordi noen linker måtte frakte mye mer pakker enn andre.

2. Ruting

1. Hva er hovedproblemet med Distance Vector Ruting (DVR) som gjør at LSR har blitt mer populær?
   - DVR har sitt count-to-infinity problem, som gjør at den veldig dårlig kan oppdage linkbrudd eller rutersvikt.
2. Hvordan har man forsøkt løse problemet til DVR?
   - Man har prøvd split-horizon tekniken. Der inkluderer man i rutingtabellen som settes sammen av avstandsmålinger og -estimeringen som man mottar fra direkte nabor også hvilken nabo som man har fått den siste estimeringen fra. Så vil en ruter A som sender en oppdateringspakke til naboruteren B slette alle de avstandsestimeringene fra oppdateringspakkene som sier at den beste ruten fører gjennom B.
3. Hvordan kan man bruke måle-trinnet i Link State Ruting (LSR) for å lage henholdsvis statiske og dynamiske ruting-tabeller?
   - Med å bruke tid som metrikk. Per linje holder rutere køer av pakker som må vente på overføring fordi linjen er opptatt. Vil man måle avstand til naboene kan man regne med tiden som målepakken venter i køen eller man kan tidsstemple pakken rett før den sendes på linjen. I den første situasjonen måler man avstand og ventetid og får en metrikk for hvor lasten på linjen forandrer seg dynamisk. I den andre situasjonen måler man den tiden som pakker trenger for selve overføringen uten ventetid, og siden

overføringsforsinkelse i trådbundene nettverk er konstant, vil dette telle som statisk metrikk.

4. Hvorfor er det ikke kritisk for nettverk som bruker LSR at ikke alle rutere er perfekt oppdatert om tilstanden til alle andre noder når rutingtabellene bygges?
   - Rutere som snakker LSR med hverandre vil bruke flooding eller andre broadcast-mekanismer for å fordele den målte avstanden fra seg selv til alle direkte naboer til alle andre rutere i nettet. Dette tar tid og kan føre til at rutere med lang avstand bygger uaktuelle rutingtabeller.
   - Dette er ikke farlig fordi pakker vil sjekkes mot mer og mer aktuelle rutingtabeller jo nærmere pakken kommer målet. Nettet må oppleve veldig rare topologi-forandringer hvis ikke denne antakelsen holder.

3. Multicast ruting

1. Hvorfor ønsker man i noen tilfeller å bruke multicast ruting?
   - Multicast gjør det mulig å spare nettverksresurser med å sende en pakke en gang fra en sender til flere mottakere, slik at rutere kan kopiere pakken så sent som mulig.
2. Gi eksempler for en multicast-ruting protokoll som baseres på DVR og en multicast-ruting protokoll som baseres på LSR.
   - Spanning tree with LSR er en multicast-rutingsmekanisme som baserer seg på LSR. Oppdateringspakker inneholder informasjon om noen endesystemer tilknyttet ruteren som rapporterer er interessert i multicastgrupper. På den måten kan alle ruterer uavhengig bestemme for hver av sine linjer om en multcast-pakke er interessent for minst en ruter som nås på en unicast-strekning gjennom denne linjen.
   - Algoritmen er bra for store nett med små grupper fordi lite traffik er nødvendig til de delene av nettet som ikke har bruk for multicast-pakkene. Multicast mekanismen PIM-SM ("protocol independent multicast - sparse mode") virker likt.
   - Reserve Path Forwarding with Pruning bruker en backwards learning algoritme. DVR brukes til å finne ut om en pakke kommer inn over en linje som er en del av den korteste veien fra den noden som har sent pakker. Hvis ikke, droppes pakken. Ellers virker algoritmen som backwards learning med å sende alle multicast-pakker som kommer inn ut på alle andre linjer, hvis ikke prune-pakker for denne linje har tidligere blitt mottatt.
   - Algoritmen er bra for nett med store gruppen siden multicast-gruppene fordeles aggressivt og raskt over hele nettet. Multicast mekanismen DVMRP ("distance vector multicast routing protocol") virker likt.

1. Nettlaget - Fragmentering/framsending

1. *Forklar prinsippet for fragmentering og reassemblering.*

   Vi må fragmentere en melding når den skal sendes ut på et pakkesvitsjet nettverk, fordi prinsippet er at alle pakker skal være av en viss størrelse for å oppnå ressursfordeling av linja. I tillegg kan forskjellige nett ha ulik kapasitet, feks Ethernet har en MTU på 1500 bytes.

Anta at maskinen som tar iniativet til kommunikasjonen benytter maksimale pakkestørrelser på (opptil) 4000 bytes og at disse kan transporteres i det nettet maskinen er koblet til. Neste nett kan kun transportere pakker med maskimal lengde på 1000 bytes. Hvordan vil fragmentering/reassemblering se ut for hvert av fragmentene i dette tilfellet?

Fragmentene vil se ut som følger: (for enkelthets skyld er ikke størrelsen av header medregnet, men pakkestørrelse satt lik datastørrelse)

Opprinnelig datagram:

Start of header			
Ident=X		0	Offset=0
Rest og header			
4000 data bytes			

Fragment Y:

Start of header			
Ident=X		1	Offset=y*1000
Rest og header			
1000 data bytes			

Forklaring til figuren: Alle fragmentene fra det originale datagrammet vil ha samme identifikator i *Ident*-feltet i headeren (unikt blant alle datagram). I dette tilfellet vil pakken på 4000B bli splittet i 4 fragment a 1000B hver når den passerer ruteren inn i nett 2. Ved splitting må det settes en *offset*-verdi som viser hvor mye av det opprinnelige datagram som har blitt sendt (y er hhv. 1000, 2000, 3000 og 4000). I tillegg brukes et M-bit (more), enten 0 eller 1, som angir om det er siste fragment, dvs alle fragment unntatt det siste har M=1. Fragmentene blir ikke satt sammen igjen før de når mottaker.

2. *Hvor er det naturlig å reassemblere fragmentene dersom det finnes flere alternativer?*

   Man kunne tenke seg at det var naturlig å reassemblere fragmentene i neste nett pakken passerer dersom MTU der er større enn MTU i nettet pakken kom fra. For å spare ressurser underveis, reassembleres det i praksis ikke før hos mottaker.

3. *Dersom et fragment blir ødelagt / borte vil det oppstå et hull i den reassemblerte pakken. Hvordan håndteres dette?*

   Hvis ikke alle fragmenter kommer frem (innen en tidsfrist), gir mottaker opp å reassemblere. Dvs, mottaker forsøker ikke å rette feil, den bare forkaster alle fragment tilhørende pakken med hull.

4. *Skisser alle de operasjoner en ruter må gjøre med en pakke fra det tidspunkt den mottar pakken på et grensesnitt (interface, port) til den sender den ut på et nytt. Beskriv også evt. feilsituasjoner som kan oppstå.*

   Først sjekker ruteren nettverksnummeret på mottakeradressen mot nettverksnummeret på alle sine nettverksadaptere (interface). Hvis match finnes, sendes pakken ut på denne linken. Hvis ikke, konsulteres forwardingtabellen. Hvis match her, sendes pakken til den ruteren som er oppført i tabellen. Ellers benyttes default ruter om en slik finnes. Siste utvei er å sende en ICMP-melding tilbake.

   I tillegg, hvis pakken er større enn utlinkens MTU, må pakken fragmenteres. Hvis fragmentering er påkrevd men ikke mulig, sendes det ICMP-melding tilbake.

2. Nettlaget - Internetworking

1. *Hva er et autonomt system? gi eksempel*

   AS er en del av Internet som er under administrasjon av en enkel enhet, feks er nettet på Ifi et AS fordi det vedlikeholdes og styres av Ifi.

2. *Hva er forskjellen på Interdomain* og *ruting? Er disse to formene for ruting helt lik, dvs basert på samme ønske om optimal rute mellom to punkter? Hvis ikke, forklar!*

   *Inter*domain ruting er prosessen å utveksle turinginfo mellom ulike ruterdomener / AS, mens *Intra*domain ruting er utveksling av rutinginfo innen et enkelt domene/subnet. AS kan selv velge hvilke rutingprotokoller de vil bruke internt.

   De har ikke samme ønske om å finne optimal rute; Intradomain ruting er ofte så vanskelig å få til pga skaleringsproblemer at målene er forholdsvis moderate. De viktigste målene er å finne en vei som ikke inneholder loop, og enhver sti i nærheten av optimal er meget bra. Fordi hvert AS kan velge ulike interne rutingprotokoller, er det umulig å regne kostnader på tvers av AS for interdomain ruting, og vi er derfor fornøyd om vi vet at en node kan nås.

3. *Internet består av en rekke autonome nett som er koblet sammen. Diskuter hvilke problem som må overvinnes for å få til ende-til-ende kommunikasjon over Internett. Beskriv vha. en tegning tjenestemodellen for sammenkobling av store nett (internett), og indiker hvilke lag i (OSI) TCP/IP-modellen som er involvert ved de ulike punktene.*

   Heterogenitet: Brukere av ulike typer nettverk skal ma mulighet til å kommunisere med hverandre, til tross for at nettene har forskjellige tjenestemodeller, adresseringsskjema og teknologi for mediumaksess *Skalering*: Internett ekspanderer med en utrolig hastighet, og denne veksten medfører følgende delproblemer: *Ruting*, hvordan finne effektiv sti i et nettverk av flere millioner noder? *Adressering*, hvordan identifisere alle nodene i nettverket?

   Når man sammenkobler nett via TCP/IP må man alltid opp på IPlaget før pakkene kan sendes videre på en fysisk link. Se figur X!

4. *Beskriv den designfilosofi som ligger bak teknologien brukt i Internett.*

Heterogene subnett skal kommunisere og opptre som ett operativt system. Sub-autonomitet; det skal ikke kreves endring i subnettene. Størs mulig adaptivitet overfor kabelbrudd, nodekræsj og trafikkbelastning. Gjøre minst mulig forutsetninger om underliggende teknologi. Skal utnytte ulike nett- og transmisjonsteknologier (konverteringsteknologi). Endesystemene skal håndtere påliteligheten.

5. *Hva menes med begrepet ?*

   Transporten av pakker skjer på raskes muli måte, uten garanti for vellyket overføring. Dette betyr at pakker kan bli borte, ankomme i feil rekkefølge, dupliseres eller bli unormalt forsinket. Det må settes en øvre grense for pakkestørrelsen.

3. Nettlaget - IP

1. *Gi en kort forklaring på hensikten til de ulike feltene i IPv4 headeren. Er noen av disse etter din mening overflødige/ mangler det noen?*

Version	HLength	TOS	Total Length			
Identification				DF	MF	Fragment offset
TTL		Protocol	Header checksum			
Source adr						
Destination adr						
Options (variabel lengde)						

2. Figuren viser IPv4-hodet. *Version* inneholder protokollens versjonsnr (4/6) og muliggjør glidende overgang og bruk av lang tid fra v4 til v6. Siden hodet kan ha variabel lengde, viser *HLength* hvor mange ord (32bit) hodet inneholder, (5-15). *Type of service* angir hvilken tjenestetype pakken ønsker. I praksis vil de fleste nåværende rutere ignorere dette feltet. *Total length* angir lengde i bytes på hele pakken (hode+data), maks 65535. *Identification* brukes slik at mottaker skal vite hvilket datagram et fragment tilhører. *DF* =1 sier at pakken ikke kan fragmenteres. *MF*=1 angir at det er flere etterfølgende fragment i datagrammet. *Fragment offset* angir hvor langt inn i datagrammet dette fragmentet skal plasseres. *TTL* er en teller som typisk dekrementeres ved hvert hopp, og hvis denne blir 0 kastes pakken (det er egentlig meningen at den skal dekrementeres flere ganger hvis pakken ligger lenge i kø, men gjøres sjelden i praksis). *Protocol* angir hvilken protokoll pakken har, feks UDP/TCP. Gyldige protokoller og deres id finnes i RFC 1700. *Header chechsum* verifiserer kun hodet. Adressene er vanlige IPadresser. Det er 5 offisielle opsjoner, hhv Security, Strict source routing, Loose source routing, record route og timestamp. Ingen av disse benyttes i stor grad. Overflødigheter og mangler avhenger av hva man ønsker seg, jfr. de endringer som har blitt gjort til IPv6.
3. *En IP adresse er på 32 bit, og det eksisterer således 2x32= 4.3milliarder unike adresser innenfor Internett. Dette tallet er i størrelsesorden jordens befolkning, og helt sikkert mye større enn tallet på alle datamaskiner på kloden. Likevel snakker man om at adresserommet til Internett er for lite. Hvordan forklarer du dette?*

IP adressene er hierarkisk oppbygd, med en nettverksdel og en verts (host) del. Videre er de delt inn i nett av klasse A, B og C hvor nettverksdelen er hhv 7, 14 og 21 bit. Fordi vi i utgangspunktet må tildele ett nettverksnummer pr potensielle nettverk, vil dette bruke opp alle adressene svært raskt, samtidig som mange nett ikke utnytter alle sine vertsadresser. Den dårlige utnyttelsen oppstår fordi vi tvinges til å utgi nettverksadresser i faste størrelser, som er svært ulike.

4. *Hva er , og hvilke problemer løser dette?*

   CIDR = *Classless Internet Domain Routing* er en metode for å aggregere ruter som behandles som en blokk med kontinuerlige klasse C IP adresser som ett nettverk. CIDR forsøker løse to av Internets skalerbarhetsproblemer: 1) at tabellene blir for store, og 2) at adresserommet blir brukt opp lenge før det er like mange maskiner som adresser på Internett. CIDR prøver balansere ønsket om å minimalisere antall ruter som en ruter må kjene til vs effektiv adresseutnyttelse. CIDR aggregerer ruter, dvs en enkelt entry i forwardingstabellen forteller hvordan mange nett kan nås. For at dette skal fungere må det deles ut blokker av klasse C adresser som deler prefix. Det lages på en måte nettverksnummer av variabel størrelse.

5. *Forklar hva er og hva slags oppgaver det skal ta hånd om. Nevn eksempel på minst èn applikasjon som benytter funksjonalitet i ICMP*

   ICMP, *Internet Control Message Protocol* brukes av rutere for å rapportere uventede hendelser, og til å teste nettverket. Det er definert 13 meldinger, og hver melding innkapsles i en IPpakke. De viktigste typene er:

   1. *Destination unreachable*, betyr typisk at ruteren ikke kan lokalisere mottakeren, eller at en pakke med DF satt ikke kan leveres videre pga lavere MTU.
   2. *Time exceeded*, vil si at TTL=0. Typisk fordi en pakke går i løkke, enorm metning i nettet, eller for lav initiell TTL.
   3. Parameter problem, betyr at det er en ulovlig verdi i et headerfelt. Skyldes vanligvis feil i programvare i en node.
   4. *Source quench*, ruteren ønsker at avsenderen skal redusere utsendingshastigheten. Benyttes sjelden i våre dager.
   5. *Redirect*, begyr at ruteren mener at pakken har blitt rutet feil
   6. *Echo request/reply*, er spørsmål og svar på om en maskin er i live
   7. *Timestamp request/reply*, er som echo men med info om transmisjonstid

   Se RFC 792 for detaljer om disse og andre typer. Ping og traceroute er to applikasjoner som benytter ICMP.

6. *Hva er , og hva benyttes protokollen til?*

   DHCP = *Dynamic Host configuration Protocol*, brukes til å tildele IP-adresser til maskiner uten manuell inngripen.

7. *Hvorfor har det blitt utviklet en ny versjon av IP? Hvilken ny funksjonalitet ligger i IPv6? Hvorfor finnes ikke IPv4's Protocol-felt i Ipv6? Må ARP-protokollen endres ved innføring av IPv6?*

pga 1) for lite adresserom og 2) manglende kontroll over tjenestekvalitet. Pga at store organisasjoner har gått over til å bruke NAT (network address translation), og dermed ikke trenger mer enn et klasse C nett til tusener av maskiner (private IP-adr som er ugyldige på Internett), er ikke argument 1) like riktig lenger.

Den nye funksjonaliteten består av: større adresserom (128 bits adr), flere adressenivåer/ruterhierarkier, økt fleksibilitet, kontroll med overførings-/tjenestekvalitet, mekanismer for økt sikkerhet (autentisering og kryptering), sanntidsanvendelser, autokonfigurering og fragmentering.

Protocol-feltet forteller destinasjonshosten hvilken protokollhandler som IP-pakken skal leveres til. Mellomliggende rutere trenger ikke denne info, så den trengs derfor ikke i hovedheader. Faktisk finnes den i IPv6 headeren, men forkledd: *next header* feltet av den siste (ekstension) headeren brukes til dette formålet

Konseptuelt er det ingen endringer i ARP, men teknisk trenger de nye IP-adressene mer plass, slik at feltene i ARP må være større.

LINKLAGET OG SUBLAYER(kapittel 3 og 4)

Oppgave 52

Hva er hovudoppgaven til Linklaget?

Oppgaven til linklaget er å sende pakker(frames) over en link mellom to noder.

Oppgave 53

Nevn noen (minst fire) typer tjenester (services) som kan være tilstede i en protokoll på link laget.

- *Innkapsling av datagram fra nettverkslaget.*
- *MAC (media access control).*
- *garanti for at paker blir sendt over linken uten feil.*
- *Feil sjekking.*
- *Feil korrigering.*
- *Heil- og halv-duplex.*
- *Strøm-kontroll (hindre buffer overflow).*

Oppgave 54

Som regel er protokoller på linklaget implementert i såkalte adaptere eller network interface cards (NIC). Hva mener vi med en adapter, og hvilke funksjoner har denne?

Et NIC adapter er et nettverkskort (interface), dette tar imot rammer (signaler på linken) og kan sende de videre til OS. NIC sender og mottar frames, det kan også behandle rammer selv.

Oppgave 55

Hvordan er det mulig å foreta feilsjekking og feilkorrigering på nettverkslaget?

For å bestemme at en ramme som ble motatt er korrekt brukes det normalt sjekksummer (CRC), den enkleste metoden for å sjekke om data ble korrekt motatt er en parity-bit sjekk, denne sjekken teller opp hvor mange bits som er satt til 1 i datastrømmen (partal = 0, oddetall = 1), og dersom denne ikke stemmer med den siste bit'en, som ble lagt til datastrømmen av forrige node er data korrupt.

Oppgave 56

Vi opererer med to typer nettverkslinker: point-to-point linker og broadcast linker. Forklar den prinsippielle forskjellen på disse, og nevn noen av de problemstillinger vi harvedstyring av broadcast linker.

Point-to-point er linker hvor data skal sendes mellom 2 noder på en link, og ingen andre. Broadcast linker er en linker hvor data kommer fra en eller flere kilder og skal til flere noder på samme link (802.11). Det som kan bli problematisk er da å finne ut når en node kan få lov til å sende data for å ungå "kræsj".

Oppgave 57

I lokale nettverk (LAN) brukes egne LAN-adresser, også kalt fysiske adresser eller MAC-adresser. Forklar kort hvordan denne adresseringen fungerer.

Hvert NIC får innbrent en MAC adresse, denne adressen skal være unik for hvert eneste NIC som blir solgt, hver produsent har fått tildelt en "gren" av adresseområdet slik kan en identifisere hvilken produsent som har laget et NIC.

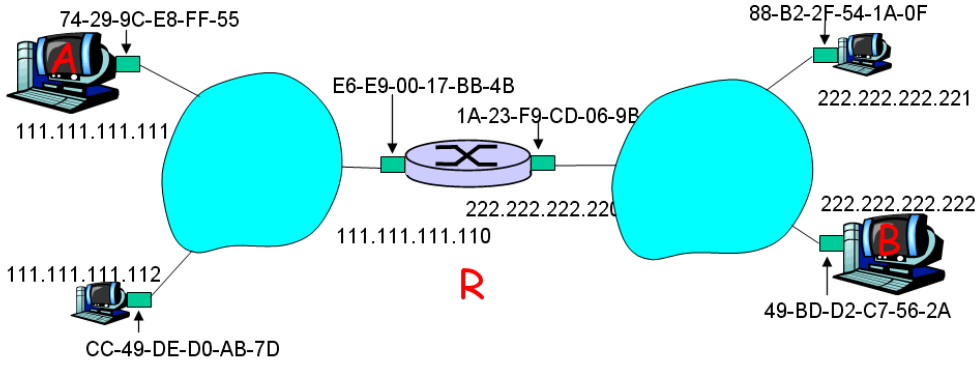
Oppgave 58

Hvordan foregår sendingen av rammer på et nettverk med broadcast linker, og hvordan brukes LAN-adressene i denne sendingen?

Hver node har en egen ARP tabell her er -adresser og MAC-adresser koblet sammen, dersom en node skal sende noe data til en annen node på en broadcast link henter den ut MAC adressen til IP som data skal til fra ARP-tabellen, deler opp pakken i rammer setter på MAC adressen og sender rammene ut på linken. Dersom den ikke har MAC adressen i sin tabell blir rammene sendt som broadcast og hver node på linken må sjekke hvilken IP den skal til, den korrekte noden vil da sende tilbake et svar på at den fikk rammene og vi kan da oppdatere ARP tabellen med den nye MAC adressen.

Oppgave 59

Beskriv kort hvordan et datagram sendes fra A til B i figuren under. Legg spesielt vekt på hvilke adresser som settes i headerene.



Pakken blir sendt fra A og skal til B:

Fra	Til	Headers
A	*R*	*Nettverks-laget:* • *Source: 111.111.111.111* • *Destination: 222.222.222.222* *Link-laget:* • *Source: 74:29:9C:E8:FF:55* • *Destination: E6:E9:00:17:BB:4B*
R	*B*	*Nettverks-laget:* • *Source: 111.111.111.111* • *Destination: 222.222.222.222* *Link-laget:* • *Source: 1A:23:F9:CD:06:9B* • *Destination: 49:BD:D2:C7:56:2A*

Oppgave 44:

Den mye brukte Ethernet-protokollen er en broadcast protokoll for lokale nettverk. Denne bruker CSMA/CD (carrier sense multiple access with collision detection). Forklar virkemåten til denne protokollen.

Opplegget er med classic Ethernet hvor det omgår at man sender data en om gangen, så hvis to eller flere stasjoner bestemmes å sende data samtidig vil det oppstå kollisjoner så hvis en stasjon oppdager kollisjon, så vil den avbryte overføringen og vente i tilfeldig tidsperiode og prøve igjen.

Oppgave 45:

Gi en kort forklaring på en hub, en gateway og en svitsj. Hva skiller disse?

En **hub** er en enhet som brukes til å koble sammen flere datamaskiner i et nettverk, for å få dem til å kommunisere med hverandre. En hub opererer på det nederste laget i OSI-modellen (fysisk lag).

En hub har to eller flere porter. Når den mottar et signal på en av portene, sender den signalet videre ut på alle porter. Dette fører til at *alle* datamaskiner som er tilkoblet huben mottar signalet. Den maskinen som har rett adresse vil da fange opp signalet og etablere kommunikasjon med den datamaskinen som sendte signalet.

Passive huber gjør ingenting annet enn å forbinde kommunikasjonslinjene sammen, mens en *aktiv* hub har en innebygget repeater som regenererer dataene som passerer for å opprettholde et sterkt signal.

Huber brukes i Ethernet og Token Ring-nettverk med tvunnet parkabel. I Token Ring-nettverk kalles en hub for en MAU (Multi-station Access Unit). Huber brukes imidlertid lite i dag. De er for det meste erstattet av switcher som er mer intelligente og bare sender signalet til adressen det er ment til.

Man trenger ikke en hub når man bare skal koble to maskiner i et nettverk, da dette kan gjøres med en spesiell type (kryssede) nettverkskabler. Som oftest er det likevel enklere å koble to eller flere pc-er sammen med en hub.

Gateway er et maskin som oppretter forbindelse mellom to eller flere nettverker og gir nødvendig oversettelse, både hardware og software.

Vi vil ikke at for lavt nivå gateway ellers vil vi ikke koble til mellom forskjellige typer av nettverk og vi vil ikke bruke for høyt nivå i gatway ellers ville koblingen fungert for enkelte applikasjoner, vi vil helst ha mellom. Kjent som nettverkslaget og ruter er en gateway som svitsjer pakker i nettverkslaget.

En **switch** eller **svitsj** er en nettverkskomponent som styrer datatrafikk mellom ulike noder i et nettverk, slik som PC, server, skriver og Internett-forbindelse. Ordet *switch* er engelsk og betyr *omkobler* eller *bryter*. Switchen opererer på lag 2 i OSI-modellen. Datatrafikk som passerer switchen blir analysert og sendt videre ut på den porten, eller i den retning, hvor mottakeren av datapakkene befinner seg. Switchen bruker MAC-adressen, også kalt den fysiske adressen, til mottakeren for å avgjøre hvor datapakkene skal.

Switcher fåes i mange varianter og størrelser fra 4 porter og opp til flere hundre porter. Portene kan ha ulike hastigheter og ulike tilkoblingstyper avhengig av behov. Det vanligste er porter for TP-kabling i hastighetene 10, 100 og 1000 MBps og porter for ulike typer fiberoptisk kabling.

En enklere variant av switch er Hub (datanettverk). En switch kan brukes på samme måte som en hub, men hub'en jobber på lag 1 i OSI-modellen og kan derfor ikke lese datapakkene og finne ut hvor de skal. En hub sender alle datapakkene ut på alle utganger i håp om at den som skal ha pakkene plukker dem opp. Denne arbeidsmetoden gjør at en hub er langt mindre effektiv enn en switch, fordi switchen kan håndtere flere datastrømmer samtidig til ulike porter, mens hub'en håndterer en strøm av gangen og pøser den ut på alle porter.

Oppgave 46:

Hvilke hovedelementer sier vi inngår i et trådløst nettverk?

Når kabler blir begrenset på hvor vi kan sette det opp og når vi vil ha fri sone for kabler.

Oppgave 47:

Hvilke utfordringer har vi ved trådløs kommunikasjon (f.eks i forhold til kablede nettverk)?

Det ligger mye hull i trådløs kommunikasjon, som sikkerhet og støy, det ligger et viss grense her. Samt begrenset distanse. Men det mest utfordrende er støy.

Oppgave 48:

MAC protokollen til 802.11 standarden bruker CSMA/CA (CSMA with collision avoidance). Forklar kort prinsipiell virkemåte til denne protokollen? Hvorfor er det viktig å bruke collision avoidance i stedet for collision detection i trådløse nettverk?

De er halvduplex som betyr at den ikke kan transmittere og lytte for stly samtidig med et enkelt frekvens. Hvis den ikke mottar støy tilbake mens den transmitterer første 64 bytes, så er rammen nesten blitt levert riktig. Med trådløs fungerer ikke kollisjons deteksjonså den unngår kollisjonen. Det er lik ethernets CSMA/CD med kanaler til å sanse før sending og eksponentiell backoff etter kollisjonen, men en stasjon som har rammer til å sende starter med en random backoff. Den venter ikke på collisjon, og antall slots til backoff er valgt fra 0, til 15. Stasjonen venter helt til kannalen er vent, ved å sanse at det ikke er signaler for en kort periode, så teller den ned ventede slots. Og venter når rammen er sent. Den sender rammen når tellern når 0. hvis rammen kommer gjennom, så skal destinasjonen sendes ved kort ack. Manglende ack er dømt til å få feil. Så senderen dobles backoff perioden og prøver igjen,

fortsetter med eksponentiell backoff som i Ethernet helt til rammen har blitt overført eller maks numer for retransmissionen har nådd.

Oppgave 49:

En 802.11 ramme har fire forskjellig adressefelt. Forklar kort hva disse feltene brukes til.

2. Linklaget - Definisjoner og funksjonalitet

a) Hva er hovedoppgaven til linklaget?

Å konvertere en rå bitstrøm tilbydd av det fysiske laget til en strøm av rammer nettverkslaget over kan benytte (og motsatt).

b) Hvilke 3 grunnleggende tjenester tilbyr linklaget?

- *upålitelig forbindelsesfri tjeneste*; avsener sender uavhengige rammer til mottaker uten kvittering (ACK)
- *pålitelig forbindelsesfri tjeneste*; avsender sender uavhengige rammer til mottaker og hver ramme kvitteres individuelt
- *pålitelig forbindelsesorientert tjeneste*; avsener og mottaker etablerer en forbindelse før dataoverføring. Linklaget garanterer at hver ramme som sendes blir mottatt, at rammene mottas nøyaktig en gang og at de mottas i riktig rekkefølge.

c) Gi en kort definisjon på 'innramming'. Hva er hensikten med innramming av data? Beskriv ulike teknikker forinnramming.

Innramming kan kort defineres som det å splitte opp en bitstrøm i håndterbare enheter. Hensikten er å legge på mottaker- og avsenderadresse, skille virkelgie data fra støy på linja, kunne oppdage og rette opp feil, samt å kunne benytte flytkontroll. Man kan brekke opp bitstrømmen i rammer (frames) ved å telle/angi antall tegn(bit eller bytes), bytestuffing med start- og stop- bytes (eks. BISYNC), bitstuffing med start- og stop-flag (eks HDLC), eller klokkebasert counting (eks SONET).

d) Gi en kort definisjon på 'transmisjonsfeil'. Hvilke metoder kan benyttes for å detektere slike feil, og hva er fordeler/ulemper ved dem?

Transmisjonsfeil kan kort defineres som de feil som kan oppstå med data når de sendes over et nettverk. Den ekstreme løsningen på feilhåndterin er å sende med en eller flere fullstendige kopier av rammen. Heldigvis finnes det en rekke teknikker som ikke sløser fullt så mye med båndbredden:
Paritet er enkelt å beregne, men kan kun oppdage 1-bitsfeil og har ingen muligheter for korreksjon av feil.
Todimensjonal paritet "fyller inn" en virtuell todimensjonal tabell med bit og beregner paritet både horisontalt og vertikalt. Dermed kan det detekteres og rettes 1 bitfeil pr

linje. Benyttes hovedsaklig i RAM og ikke så mye i kommunikasjon siden det er vanskelig å beregne on-the-fly.
Sjekksum kan betraktes som en videreutvikling av paritet hvor man beregner summen av bitene i rammen og sender dette tallet modula et nøye valgt tall t. Denne teknikken kan oppdage et antall bitfeil avhengig av størrelsen på t (som angir antall bit som legges i rammen).
CRC er pålitelig og effektiv og den mest brukte teknologien (med variasjoner) innenfor kommunikasjon. Den er tyngre å beregne enn paritet, men likevel lite CPU-krevende i forhold til hva moderne maskiner har tilgjengelig. CRC beregnes mens bitene sendes ut, og er relativt sterk i forhold til mengden redundante data.
FEC er pålitelig og gir mulighet for å rette feil på bekostning av behov for mer redundante data enn andre ikke-rettende teknikker. Benyttes når det er mer effektivt å rette enn å retransmittere (som ved høy RTT), eller ved konstant datstrøm (audio/video) hvor retransmittering ikke er aktuelt

1. Linklaget - Funksjonalitet forts.

- a) Forklar prinsippet bak CRC og FEC, ta med hva forkortelsene står for. Hvilke faktorer påvirker beslutningen om å korrigere feil ved hjelp av CRC-metoden for deteksjon kombinert med retransmittering av forkastede pakker vs bruk av FEC? Hvorfor legges CRC-sjekken nesten alltid sist i rammen på linklaget?

  Ideen bak begge teknikkene er å oppdage feil, forskjellen ligger i at mens CRC kun detekterer, så kan FEC også rette feil til en viss grad. *Cyclic Redundancy Check* er en polynomisk kode, basert på å behandle bitstrenger som representasjoner av polynom med kun koeffisientene 0 og 1. En sekvens av k bit, er således listen over koeffisienter for et polynom av grad k-1, eks 110001 gir x^5+x^4+1 (husk at x^0 = 1). Polynomisk aritmetikk utføres modulo 2, både addisjon og subtraksjon tilsvarer bitvis XOR. Partene enes om et generatorpolynom G(x) med grad r. For å sende en melding med m bit, legg til r antall 0 i den minst signifikante enden av opprinnelig bitsekvens. Utfør polynomdivisjon av den utvidede sekvensen med G(x), eksempel i Tanenbaumboka. XOR inn resten fra divisjonen i den utvidede sekvensen (på plassen der de ekstra 0bitene ble lagt til). Resultatet er T(x) som sendes over linken. Hvis mottaker kan utføre divisjonen T(x) på G(x) og få 0 i rest, er overføringen feilfri. Mengden feil CRC kan oppdage er avhengig av polynomet.

  *Forward Error Correction*benytter seg av å sende redundant info sammen med data. FEC er således egentlig ikke èn teknikk, men en samling teknikker, hvor den enkleste formen er å sende med koplett duplikat av originaldata. De feilrettende egenskapene avhenger av Hamming distansen (antall bitposisjoner hvor to kodeord (m databit + r redundante bit) er ulike). For å oppdage d feil, trengs en distanse kode d+1, fordi det da ikke er noen måte at d enkle bitfeil kan endre et gyldig kodeord til et annet gyldig kodeord. For å rette opp d feil, kreves imidlertid distansekode 2d+1 fordi da er kodeordene så lang fra hverandre at selv ikke d feil kan blande sammen to gyldige kodeord. se side 193-195 i Tanenbaum.

  Faktorer som påvirker valget er: linkens kvalitet(hyppig forekommende 1-bitsfeil peker i retning av FEC), linkens RTT(ved lav RTT bruk CRC og retransmisjon hvis

mulig, ellers FEC hvis kostnaden ved retransmisjon blir for høy eller du bare har ett forsø pr ramme). Husk bevaring av rammerekkefølge.

For å få sendt data ut på linken så raskt som mulig, er det praktisk å regne ut CRC-koden mens bitene sendes (on-the-fly), og sette koden sist i rammen. Hvis koden skal settes i header, må hele rammen først buffres mens koden beregnes, noe som krever både mer tid og bufferkapasitet, og øker forsinkelsen gjennom nettet.

- b) En bitsekvens 10011101 overføres ved å benytte standard CRC som beskrevet i læreboka. Generatorpolynomet er x^3 +1. Vis hvilken bitsekvens som overføres. Anta at det tredje bitet fra venstre inverteres under overføringen, og vis så at denne feilen oppdages på mottagersiden.

  Polynomet blir 1001, og vi hekter på tre 0bit i den minst signifikante enden av bitsekvensen og får 10011101000. Den utvidede bitsekvensen divideres så på polynomet og vi får rest 100 som XORes inn i bitsekvensen slik at det som overføres blir 10011101100. Sekvensen som mottas med feil er således 10111101100, og ved divisjon med polynomet gir dette rest hos mottaker på 100 som er ulik 0, hvilket betyr at feil er oppdaget og retransmisjon kan bli trigget.

- c) Gi en kort definisjon av begrepet 'flytkontroll'. Hvordan håndteres flytkontroll på linklaget?

  *Flytkontroll*regulerer trafikken over en kanal ved å gjøre det mulig for mottaker å justere hvor mye data den til enhver tid er i stand til å motta. Dette kan oppnås ved at mottaker sender en egen melding som ber sender stanse eller redusere raten den sender data med. Sliding window mekanismen kan også benyttes til dette ved at mottaker ikke sender ACK når den ikke har ledig kapasitet. Riktignok vil dette resultere i unødvendige retransmisjoner, men på linklaget kan linken uansett ikke benyttes til noe annet mellom et gitt sender-mottaker par når mottaker er opptatt.

- d) Forklar virkemåten til 'Stop-and-wait' (SAW). Hvordan håndteres feilsituasjoner som feks. tapte/forsinkede rammer? Hva slags funksjonalitet tilbyr denne protokollen?

  SAW sender ut en pakke og venter så på kvittering (ACK/NACK) før den sender neste pakke. Feilsituasjoner løses ved timeout: hvis kvittering (ACK) ikke er mottatt når tiden utløper, retransmitteres pakken. Ved bitfeil sendes enten NACK, eller så kastes pakken stille og man venter på at timeout skal trigge retransmisjon. Denne protokollen besørger pålitelig overføring over linken samt bevaring av pakkerekkefølgen.

- e) En kanal har en bitrate på 4kbps og propagasjonsforsinkelse på 20ms. For hvilket område av rammestørrelser gir SAW en effektivitet på minst 50%?

  Effektiviteten vil være 50% når tiden det tar å overføre rammen er lik RTT (propagasjonsforsinkelse). Ved en overføringshastighet på 4 bit pr millisekund, vil en kunne sende 160 bit på 40ms (RTT, 2x enveis delay). For rammestørrelse over 160bit vil derfor SAW være rimelig effektiv.

- f) Forklar virkemåten til 'Sliding window'. Hva slags funksjonalitet tilbyr denne protokollen? Redegjør kort for problemstillinger knyttet til sekvensnummer i Sliding window

  *Sliding window*sender pakker og venter ikke på kvittering for hver enkelt pakke, men tillater et gitt antall pakker utestående samtidig før ACK må mottas. Se side 211 i Tanenbaum. Denne protokollen besørger, på samme måte som SAW, pålitelig overføring over linken og bevaring av pakkerekkefølge, men i tillegg har sliding window også støtte for flytkontroll. Hovedproblemet med sekvensnummer i sliding window protokollen er at man må sørge for at det ikke er overlapp mellom sekvensnummer for å kunne skille mellom nye og duplikate pakker. For *selectiv repeat* løses dette ved å benytte et sendevindu som er mindre enn (max sekvensnr +1)/2. Når båndbredden øker kan sekvensnummerintervallet bli for lite.

2. Ethernet - Egenskaper

- a) Beskriv kort den historiske utviklingen av Ethernet slik vi kjenner det i dag. Hva er forskjellen på de facto standarden Ethernet og ISO's CSMA/CD standard? Er det mulig for dem å sameksistere i et nett, og i så fall hvordan?

  Stikkord for utviklingen: Norman Abramson, Hawaii ALOHANET, Bob Metcalfe, Ethernet 1976, coax, multidrop cable, 2.94Mbps, DIX standard 1978 10Mbps, IEEE802.3 1983
  Forskjellen på standardene ligger tolkningen av rammens 2 byte felt mellom adresser og data. de facto standarden bruker dette som et Type-felt som forteller mottaker hva som skal gjøres med rammen, dvs hvilken prosess den skal leveres til. IEEE-standarden som ISO benytter tolker dette som et lengdefelt. Siden payload i den første standarden var 1500B eksakt, og mindre datamengder måtte paddes til å fylle ut disse 1500, så kan de to versjonene sameksistere ved at Eternet standarden ikke benytter type verdier under 1500, og 802.3 kan angi lengder opptil 1500.

- b) Redegjør for hvordan et Ethernet er bygd opp og hvilke fysiske begrensninger som gjelder. Hvilke rammer kan et ethernet adapter motta?

  Ethernet benytter CSMA/CD teknologi, se oppgave 3 for detaljer. Dette innebærer at det er et delt medium hvor det kan forekomme kollisjoner hvis flere stasjoner forsøker sende data samtidig. Fakta: 48bits adresser, rammene må inneholde minst 46B og maks 1500B data, rammene har en 32bits CRC, standarden er generelt bitorientert, det benyttes coaksialkabel med typisk impedans 50 ohm, minimum 2.5 meter mellom hver host, maks 1024 hoster, maks 4 repeatere mellom 2 vilkårlige hoster, maks utstrekning 2500 meter når det benyttes 4 repeatere, hvert segment kan være på maks 500meter.

  Vanligvis vil et ethernet adapter ta imot rammer som har mottaker adresse som enten matcher det enkelte adapters egen adresse, eller en kringkastingsadresse. Det er også mulig å instruere adapteret til å ta imot ulike multikastaderesser, eller også til å fange opp alle rammer, noe som kalles promiscous mode.

- c) Ethernet er også implementert for datarater på 100Mbps og 1Gbps. Beskriv eventuelle problemer med å øke bitraten fra 10Mbps til disse, og forklar mulige løsninger på problemene.

Ved å øke senderaten fra 10 til 100 Mbps, må også kravene til transmisjonsmediet økes. Legging av ny kabel er kostbart, så man ønsker å benytte eksisterende kabler så sant det lar seg gjøre. Kollisjonsdeteksjonsalgoritmen er basert på at enhver stasjon skal kunne oppdage en kollisjon mens en stasjon sender ut en ramme. For 10Mbps Ethernet med maksimal utstrekning 2500m, og dermed minimums rammestørrelse 512bit, har hvert bit en utstrekning på 100ns. Ved å øke raten til hhv 100Mbps og 1Gbps, reduseres utstrekningen til hhv 10 el. 1 ns. Dvs at hvis man skal holde utstrekningen konstant, vil delay*bandwidth produktet øke med en faktor 10 / 100.

For 100Mbps er dette løst ved å kreve at det brukes TP- eller fiber-kabler. For TP gjelder da kat3 med 4 par og maks-segment på 100m, eller kat5 med 2 par og 200m utstrekning. Fiberkabelens utstrekning her er satt til maks 2000. I praksis implementeres 100Mbps med kat3. Alle slike systemer benytter hubs, enten shared eller switched (i det siste tilfellet har hver stasjon sitt kollisjonsdomene). For å kunne detektere kollisjoner med en shared hub kan man enten redusere maks utstrekning slik at 512bit igjen er nok til å fylle mediet, eller å øke minimum rammestørrelse, hvilket ikke er særlig praktisk med tanke på sameksistens av systemer.

3. Ethernet - CSMA/CD

Forklar virkemåten til CSMA/CD protokollen. Hva forbindes med 51.2 microsec? Hva ligger i følgende begrep: (non)persistent, kollisjonsvindu, eksponensiell back-off

Carrier Sence Multiple Access with Collision Detection
Mottakersiden av et ethernet adapter er relativt enkelt, det er senersiden som inneholder kompleksiteten. Algoritmen er som følger: hvis mediet er ledig kan enhver stasjon begynne å sende umiddelbart. Siden protokollen tillater maks 1500B data + 27B header (12216 bit) vil en sender ved 10Mbps legge beslag på mediet i ca 1.2ms Etter transmisjon må adapteret vente 51.2microsec før neste ramme kan sendes for å forhindre at et enkelt adapter kan beslaglegge mediet kontinuerlig. I denne pausen har andre adaptere som venter, en sjanse til å starte sin sending. Med en maksimal utstrekning på 2500m og 10Mbps, er delay*bandwidth produktet 512 bit (14B header, 46B data, 4B CRC). Dersom flere stasjoner starter å sende data ut på mediet i løpet av et intervall, *kollisjonsvinduet*, vil samtlige stasjoner oppdage dette og avslutte transmisjonen etter at minst 512 bit er sendt. Disse 512 bitene fyller vinduet og sikrer at alle stasjoner ser kollisjonen, og det er også derfor et adapter må vente den spesifikke pausen mellom rammer.

Etter at en kollisjon er oppdaget, og alle involverte sendere har trukket seg tilbake, benyttes en eksponensiell backoff algoritme for å avgjøre når det muligens er trygt å starte sending igjen. Formelen er 2^n * 51.2microsec, hvor n er et tilfeldig tall mellom 0 og det antall kollisjoner som hittil har forekommet for den gitte rammen. Dvs at ventetiden for det første alltid er et multiplum av kollisjonsvinduet, for det andre potensielt øker for hver kollisjon som skjer, og for det tredje ikke endres likt for alle stasjoner slik at sannsynligheten for at èn av partene skal klare å sende uforstyrret øker for hver kollisjon som oppstår. *k-persistent* betyr at stasjonen begynner å sende med det samme mediet blir ledig med en sannsynlighet k. *non-persistent* betyr at stasjonen ikke lytter kontinuerlig på mediet for å finne ut når det blir ledig, men istedenfor sjekker linjen på tilfeldige tidspunkt, og hvis den da er ledig så startes transmisjon.

MULTIMEDIA KAPITTEL 7

Oppgave 50:

Hva er typiske karakteristika for multimedia i nettverk?

Oppgave 51:

Hvilke 3 klasser av Multimedia applikasjoner snakker vi om (nevn eksempler) ?

Streaming
- Klienter forespørsel audio / video-filer fra servere og rørledning mottak over nettverket og display
- Interaktivt: Brukeren kan styre driften (tilsvarende til VCR: pause, fortsette, spole fremover, bakover, etc.)
- Forsinkelse: fra klienten forespørsel til displayet start kan være 1 til 10 sekunder
- Eksempel: RealAudio / RealVideo

Unidirectional Real-Time:
- lik eksisterende TV-og radiostasjoner, men levering på nettverket
- Ikke-interaktiv, bare lytte / visning
- Eksempel, online kurs kringkasting

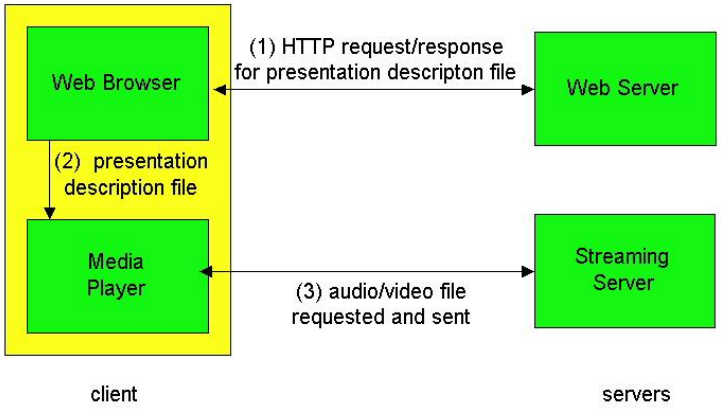
Interactive Real-Time :
- Telefon samtale eller videokonferanse
- Strengere krav forsinkelse enn Streaming og
Einsretta grunn av interaktive sanntids natur
- Video: <150 msek akseptabel
- Audio: <150 msek bra, <400 msek akseptabel

Oppgave 52:

Gitt begrensningene vi har i dag på internett (best-effort), hva kan gjøres for å kunne håndtere multimedia så godt som mulig ?

Bare legg til mer båndbredde og forbedre caching
evner (over-forsyning)!
To Camps
- Trenger stor endring av protokollene (Integrated
Services):
• Innlemme ressurs reservasjoner (båndbredde, prosessering,
bufring), og ny planlegging politikk
• Sett opp serviceavtaler med programmer, overvåke
og håndheve avtaler, belaster deg tilsvarende
- Trenger moderate endringer ("Differensiert Services"):
• Bruk to trafikk klasser for alle pakker og differensiere
service tilsvarende
• Lad basert på klasse av pakker
• kapasiteten i nettet for å sikre førsteklasses pakker
ikke medfører noen betydelig forsinkelse på rutere

Oppgave 53:



Figuren over viser en typisk løsning som benyttes ved Streaming Stored Video/Audio. Forklar gangen i fra klienten henter opp en side som inneholder streamed multimedia data, til video/audio blir avspillt til brukeren ?

1. *Browser spør webserver etter metadata (informasjon om det som skal avspilles, her informeres det om båndbredde og komprimeringsteknikk som er benyttet).*

2. *Browser starter multimedieavspilleren, og overfører metadata. Avspilleren setter opp riktig dekoding og kobler opp mot streamingserveren iht. Protokoll som er spesifisert i metadata.*

3. *Streamingserver kommuniserer nå direkte mot avspilleren og overfører video/audio.*

Oppgave 54:

Hvilke utfordringer har vi ifm. sanntids interaktive mulitimedia applikasjoner (Real Time Interactive Applications) som f.eks. Internet telefoni ?

Internet phone applications generate packets during talk spurts

Bit rate is 8 KBytes, and every 20 msec, the sender forms a packet of 160 Bytes + a header to be discussed below

The coded voice information is encapsulated into a UDP packet and sent out; some packets may be lost; up to 20 % loss is tolerable; using TCP eliminates loss but at a considerable cost: variance in delay; FEC is sometimes used to fix errors and make up losses

End-to-end delays above 400 msec cannot be tolerated; packets that are that delayed are ignored at the receiver

Delay jitter is handled by using

timestamps, sequence numbers, and
delaying playout at receivers either a fixed
or a variable amount
With fixed playout delay, the delay should
be as small as possible without missing
too many packets; delay cannot exceed
400 msec

Oppgave 55:

Hvordan prøver man å løse problemet i forrige oppgave?

Prøvd å fikse dette med playout delay. Dvs!

Målet er å bruke en verdi for pr som sporer
nettverksforsinkelser ytelse som varierer i løpet av en
telefonsamtale
Den playout forsinkelsen er beregnet for hver diskusjon sprute
basert på observert gjennomsnittlig forsinkelse og observerte
avvik fra dette gjennomsnittet forsinkelsen
Beregnet gjennomsnittlig forsinkelse og avvik av gjennomsnittlig
forsinkelse er beregnet på en måte som ligner
estimater av RTT og avvik i TCP
Begynnelsen av en diskusjon sprute er identifisert fra
undersøke tidsstempler i påfølgende og / eller
sekvens antall biter

Konkrete protokoller

RTSP – Real Time Streaming Protocol

Oppgave 56:

Gi en kort beskrivelse av RTSP?

For brukeren kan kontrollere vise: spole fremover,
pause, fortsette, osv. ...
Out-of-band-protokollen (bruker to tilkoblinger, en
for kontroll meldinger (Port 554) og for media
stream)
RFC 2326 tillater bruk av enten TCP eller UDP for
kontrollen meldinger tilkoblingen, noen ganger
kalt RTSP Channel
Som før, er meta-fil kommunisert til web
nettleser som deretter lanserer Player; Player

setter opp en RTSP-tilkobling for kontroll
meldinger i tillegg til tilkoblingen for
streaming media

RTP – Real Time Protocol

Oppgave 57:

Gi en kort beskrivelse av RTP ?

bruker UDP
l definerer format av tilleggsinformasjon som kreves av søknaden
(sekvensnummer, tidsangivelser)
l bruker en spesiell sett med meldinger (RTCP) å utveksle periodiske rapporter
l en RTP økt, en medium flow.

Gir standard pakke format for real-time-programmet
o Angir header feltene nedenfor
o Payload Type: 7 bits, som gir 128 mulige forskjellige typer
av koding, for eksempel PCM, MPEG2 video, etc.
l ulike medier er ikke multiplekset
o Sekvens Antall: 16 bits; tilfeldig tall økes med
en for hver RTP datapakker sendt; brukes til å oppdage pakketap

Tidsstempel: 32 byte, gir prøvetaking instant av den første audio / video
byte i pakken, brukes til å fjerne jitter introdusert av nettverket
l klokkefrekvens avhenger av søknader
l tilfeldig opprinnelig verdi
l flere pakker kan ha like tidsstempler (eg. samme video ramme), eller til og med i
lidelse (eg. interpolert rammer i MPEG)
o Synkronisering Kilde identifikator (SSRC): 32 bits, en id for
kilden til en strøm; tildelt tilfeldig av kilden
o Diverse felter: Bidra Source identifikator (CSRC)
Transport Layer
7
Type av nyttelast
o Audio
l PCM A-lov
l PCM m-lov
l GSM
o Video
l CelB
l JPEG
l H.261
l MPEG
Transport Layer
8

RTCP – Real Time Control Protocol

Oppgave 58:

Forklar hva RTCP er og hva den brukes til ?

RTP Control Protocol (RTCP)
o Protokoll angir rapporten pakker utveksles mellom kilder og destinasjoner av multimedia informasjon
o Tre rapporter er definert: Receiver rapport (RR), Sender rapport (SR), og Kilde beskrivelse (SDES)
o Rapporter inneholder statistikk som antall pakker sendt, antall pakker tapt, inter-ankomst jitter
o Brukes til å endre avsender dataoverføringshastigheter og for diagnostikk formål

Hvis hver mottaker sender RTCP pakker til alle andre mottakere, trafikken belastning som fremkommer kan være store
o RTCP justerer intervallet mellom rapporter basert på antall deltakende mottakere
o Vanligvis begrense RTCP båndbredden til 5% av økten båndbredde, delt mellom avsender rapportene (25%) og mottakere rapporter (75%)

Funksjoner
l overvåke nettverket QoS
- flytkontroll og metningskontroll
l identifisering av deltakere
- vedvarende id (CNAME = Canonical Name)
l bestemme antall deltakere
l session informasjon
l trafikk av RTCP <5%
o Format av RTCP pakker
l SR: sender rapporter
- informasjon om kilden
- source statistikk
l RR: mottak rapporter
- mottaker statistikk
l SDES: source beskrivelse
- CNAME
l BYE: slutten av deltakelse
l APP: program spesifikke funksjoner

SIP

Oppgave 59:

Gi en kort beskrivelse av SIP (Session Initiation Protocol)? Hva er en SIP proxy

Beskriv gangen i en SIP-basert oppkobling

Lett generisk signaleringsprotokoll
• Internett-telefoni og videokonferanser
- Ring: assosiasjon mellom antall partcipants
- Signalering forening som signaliserer staten på endepunkter (uten nettverk ressurser)
• Flere "tjenester" trengte
- Navn translaton, brukerens plassering, funksjon forhandling, ring kontroll

Etablere samtaler mellom brukere
- Direkte eller videresending (manuell og automatisk)
- Re-forhandle ringe parametre
- Avslutte og overføre samtaler
• Støtter personlig mobilitet (endring av terminal)
- Gjennom fullmakter eller omdirigering
• Kontroll, beliggenhet og media beskrivelse (via SDP)
• Extensible
- IMS - Internet Multimedia Subsystem - neste generasjon av telekom 'service gateways

Grunnleggende metoder:
- INVITE: session setup - som RTSP SETUP og beskrive i ett
- ACK: liker RTSP ACK
- VALG: liker RTSP ALTERNATIVER
- BYE: avslutte en økt
- AVBRYT: avslutte en pågående økten drift
- REGISTER: registrere seg på et sted server, oppdatering beliggenhet, ...
• Ekstra Metoder (delvis standardisert):
- INFO: frakter informasjon mellom User Agents
- SE: be noen om å sende en invitasjon til en annen deltaker
- Abonner: forespørsel om å bli varslet om spesifikk hendelse
- VARSLE: varsel om spesielt arrangement

Proxy fremover forespørsler
- Eventuelt parallelt til flere verter
- Kan ikke akseptere eller avvise anrop
- Nyttig å skjule plasseringen av callee

Physical Layer

Referanse: Kap 2

1. Funksjoner - Fysisk lag

a) Hva er oppgavene til lag 1 i OSI modellen ?

- å klokke ut og ta imot bit, multipleksing/demultipleksing, koding av bit i basisbånd, modulasjon av bit inn på en bærebølge (amplitude, frekvens og fase -modulasjon)

b) Redegjør for problemstillinger rund sending av digitale signaler og synkronisering av sender og mottaker.

Digitale signaler sendes ved bruk av diskrete signaler (høyt og lavt signal). Utfordringen ligger i koding av bit innpå mediet:

- Hvordan vite om det sendes mange påfølgende 0-bit eller om linja er død?
- Forskyving av 'baseline' signalet (gjennomsnittsverdi for høyt/lavt signal)
- Vanskelig klokkeavledning hvis mange påfølgende like signal. Klokkene hos sender og mottaker må være nøyaktig synkronisert for at mottaker skal kunne lese av samme bitverdi som ble sendt fra den ander siden, og også lese av like mange verdier

c) Redegjør for forskjellige kodingsmetoder som skal avhjelpe problemene nevnt i oppgave b)

Manchesterkoding (kap4)

d) Gi eksempler på forskjellige transmisjonsmedia. Diskuter fordeler og ulemper ved dem, og angi anvendelsesområde.

twisted pair - billig, og det mest brukte mediet. Hovedsaklig brukt i telenettet og LAN i bygninger.
coaxialkabel - kan bære mange typer signal samtidig (data, tv, osv) over korte avstander. Brukes til overføring av TV-signaler og i LAN (data 50 ohm, tv 75 ohm)
optisk fiber - har stor kapasitet / høy båndbredde; er mindre og lettere enn kobberledninger (twisted pair); har lavere demping av signalstyrke og denne dempingen er konstant over en lengre rekkevidde; elektromagnetisk isolasjon (påvirkes ikke av andre elektromagnetiske kilder pga bruk av lysbølger og er dermed vanskeligere å avlytte/bryte); kan legges med større avstand mellom repeatere. Pga disse fordelene er fiberkabler svært mye brukt.
terrestrial mikrobølge - trådløst medium som krever at sender og mottaker har "øyekontakt". Er et alternativ til koaks eller fiber og benyttes steder der det er vanskelig å legge kabel.
satellitt mikrobølge - trådløst medium med rettede antenner som gir stort footprint som gjør det velegnet til distribusjon. Brukes til TV, telefoni og organisasjoner med private linker.
broadcast radio (radiobølger) - trådløst medium som ikke krever rettede antenner. Brukes til vanlig radio og TV
infrarød - trådløst medium hvor sender og mottaker må ha direkte øyekontakt og signalene stanses av bl.a. vegger, noe som gir bedre sikkerhet og minimerer

interferensproblemer (som mikrobølger har). Ingen frekvensallokering av det infrarøde spekteret. Brukes mellom apparater som er fysisk nær hverandre, etc fjernkontroller, mobiltelefon, PC)

e) Redegjør for Nyquists og Shannons teorem. Forklar hvordan Nyquists teorem knyttes til begrepet sampling. Gjørnoen egne forutsetninger og gi et regne-eksempel.

Kanalkapasiteten til en link påvirkes av en rekke forhold: datarate(bps), båndbredden til signalet gitt av transmisjonsmediet (Hz), gjennomsnittlig støynivå på linken og feilraten. Ønsket er å utnytte kommunikasjonslinken maksimalt, men alle medier har en båndbreddebegrensning gitt av mediets fysiske egentskaper og evt. bestemmelser ang. bruk (frekvensallokering, signalstyrke etc).

Nyquists teorem gir svar på hvor stor kanalkapasiteten målt i bps er ved støyfrie (ideale) forhold. C=2WlogM bps
Hvor C er teoretisk maksimal kanalkapasitet, W er signalets båndbredde og M er antall signalnivå (man tar binærlogaritmen av denne). Antall signalnivå beskriver hvor mange verdier signalet kan ha, dvs for 2 nivå kan vi kode bitverdiene 0 og 1. Ved flere nivåer (finere inndeling av spenningsskala) vil feks hvert enkelt av 4 nivå betegne 2 bit. Jo flere nivå, jo vanskeligere er det for mottaker å skille nivåene og avgjøre hvilket nivå et gitt signal tilhører.

Nyquists teorem uttryker en matematisk egenskap og har ikke noe å gjøre med teknologi. Det sier at hvis du har en funksjon hvis Fourier spektrum ikke inneholder noen sinus eller cosinus over *f*, så vil du ved å sample funksjonen ved en frekvens på 2*f* fange all tilgjengelig informasjon. Det gjelder således for alle media.

Shannons teorem tar utganspunkt i forholdet mellom datarate, støy og feilrate. Teoremet sier at for et gitt støynivå vil høyere datarate medføre flere feil. Det viktige parameteret i formelen er signal-støy forholdet (S/N), dvs forholdet mellom styrken i signalet og styrken på støyen. Det er hensiktsmessig å uttrykke dette forholdet i desibel fordi signalstyrken ofte faller logaritmisk. Måles ofte hos mottaker: (S/N)db = 10log(signalstyrke/støystyrke).
Dette forholdet setter en øvre grense for den oppnåelige dataraten, og kanalkapasiteten er gitt av formelen C = W log (1+S/N) bps
hvor kanalkapasiteten er produktet av båndbredden (Hz) og binærlogaritmen til 1+S/N

Regneeksempler:
Nyquist: M=8, W=3100Hz gir C = 18600 bps (støyfritt)
Shannon: W = 3100Hz, S/N = 30db gir C=30.894 bps (med støy)

Prinsipper og begreper

1. Kommunikasjonsformer

Gi en kort definisjon på følgende begrep:

- a)

linjesvitsjing er en teknikk som tradisjonelt har vært brukt i telefonnettet, men som nå også benyttes til dataoverføring. Tre faser inngår i kommunikasjonen: 1. oppsett av linje mellom avsender og mottaker vha. fysiske eller elektroniske svitsjer 2. overføring / transmisjon 3. nedkobling av linje. Fordelen ved linjesvitsjing er at man får en egen "fysisk" forbindelse reservert til den prosessen som foretok oppsettet, men på bekostning av å ha uutnyttet ressurs i nettet når denne prosessen ikke benytter linja.

pakkesvitsjing innebærer at data som skal overføres, først deles opp i pakker som så sendes ut på de fysiske linjene. Hver pakke inneholder 'hode' og 'hale' i tillegg til informasjonsbærende data. Hvilken sti pakken skal ta gjennom nettet, bestemmes av adresseinformasjonen som ligger i den enkelte pakkens hode. I hver svitsj mellom avsender og mottaker foretas et valg angående hvilken retning pakken skal sendes basert på analyse av hode og den generelle øyeblikkstilstanden i nettet. Ulike pakker kan dermed følge ulike stier gjennom nettet mellom samme avsender og mottaker. Fordelen med pakkesvitsjing er at flere prosesser kan dele den samme fysiske linjen i nettet, og dermed utnytte ressursene bedre. I tillegg åpner pakkesvitsjing muligheten for å rute utenom linker eller noder som er nede eller overbelastet. Passer bra for asynkron trafikk.

- b)

  synkron overføring (transmisjon) vil si at pakkene ankommer mottaker i samme takt som de leveres av sender, dvs at forsinkelsen er konstant mens overføringen foregår. eks tale- og video-overføring

  asynkron overføring vil si at forsinkelsen varierer og at avsender og mottaker ikke er i takt. Pakkesvitsjing impliserer asynkron overføring, og her kan også rekkefølgen av pakkene innbyrdes endres underveis fordi de muligens følger ulike stier, og pakkene må dermed sorteres ved mottak. eks filoverføring, interaktiv kommunikasjon etc

- c)

  i>forbindelsesfri (datagram) overføring kan sammenlignes med å sende et brev i det vanlige postsystemet. En pakke (datagram) utstyres med en fullstendig adresse som gjør det mulig å rute pakken frem til mottaker uten at det settes opp noen forbindelse på forhånd

  forbindelsesorientert overføring kan sammenlignes med det å koble opp en telefonsamtale eller en TCP-forbindelse: Oppkobling - bruk - nedkobling

- d)

  multipleksing i telenett vil si å gjøre bruk av noder i nettet (såkalte multipleksere) for å muliggjøre for flere linjer med lav bitrate å dele en linje med høy bitrate. Dataoverføringskapasiteten på den sistenevnte linja er større enn eller lik summenav bitratene for hver av lavkapasitetslinjene.

2. Typer av nettverk

- a) Gjør rede for forskjellen mellom punkt-til-punkt nett og kringkastingsnett.

Punkt-til-punkt nett: muliggjør dataoverføring mellom én avsender og én mottaker om gangen. Nettet består av mange forbindelser mellom individuelle par av maskiner. For å komme fra kilde til destinasjon kan en pakke i denne typen nett måtte gå innom mange mellomliggende maskiner (svitsjer). Det er viktig å finne en optimal rute gjennom nettet. Siden det her dreier seg om sender-mottaker par, kalles denne typen overføring gjerne 'unicast'.

Kringkastingsnett har gjerne en felles kommunikasjonskanal som deles av alle maskinene på nettet. Når en maskin sender en pakke ut på nettet, vil denne derfor mottas av alle de andre maskinene, men kun den eller de maskinene pakken er adressert til, vil lese den inn i sin hukommelse. Pakker som ikke er adressert til en bestemt maskin, blir ignorert av den maskinen. Denne typen nett muliggjør to nyttige kommunikasjonsmåter, nemlig 'kringkasting' (broadcast) og 'multikasting' (multicast). I det første tilfellet vil en pakke som blir sendt ut på nettet bli plukket opp og prosessert av samtlige maskiner i nettet, mens i det andre tilfellet er det kun et subsett av maskinene som vil gjøre dette.

- b) Hva kjennetegner *Internet* og hvilke funksjoner må dette nettet kunne håndtere?

  For det første består Internet av en meget stor samling autonome nett, dvs nett som i utgangspunktet er laget for å betjene en gruppe brukere (eks lokalnettet ved UiO). Hvert av disse (sub-)nettene fungerer uavhengig av hverandre, men er sammenknyttet vha. ulike typer langdistansenett eller rene høyhastighets forbindelser. Internettprotokollen (IP) binder alle de autonome nettene sammen til Internet ved å bl.a. tilby universell adressering. Sentrale funksjoner som må håndteres er bl.a. ruting på Internet-nivå mellom subnett (globale IP-adr), ruting på subnett-nivå mellom "lokale" maskiner (lokale subnett-adr), fragmentering og re-assemblering.

- c) Beskriv de mest vanlige LAN topologiene

  Bus : alle maskinene er koblet til en lineær kabel, dvs uten sammenkoblede ender. Benyttes for Ethernet-baserte lokalnet, hvor protokollen tar høyde for at det vil oppstå kollosjoner på kabelen når to eller flere maskiner forsøker sende samtidig.

  Ring : kabelen maskinene er tilkoblet utgjør en sluttet ring. LAN protokoller som benytter ring-topologier har sikre mekanismer som hindrer at kollisjoner oppstår (eks Token ring, FDDI)

  Stjerne : Maskinene er tilknyttet en felles enhet /svitsj) som sørger for at data kan sendes til riktig mottaker (maskin eller nett) hvis denne også er tilkoblet svitsjen (eks Ethernet svitsj).

3. Adressering og ruting

- a) Nevn de forskjellige typene adressering, og gi eksempler på applikasjoner hvor de benyttes

  Unikast, adresse til en enkelt maskin (MSN)
  Kringkasting, fellesadresse for alle maskiner i nettet (melding til alle)
  Multikast. gruppe-kringkasting / gruppe-adr (mailingliste, deltagere i videokonferanse)

- b) Hvilke nødvendige egenskaper bør en nettverksadresse ha?

  En nettverksadresse bør ha stort nok adresserom og gi en entydig identifikasjon av maskiner tilknyttet nettet. Som oftest vil hierarkisk ordenede adresser være å foretrekke. Videre bør det være mulig med dynamisk tilordning av adresser for enkelt vedlikehold av nettet. Adressestrukturen og navnehåndteringen bør også muliggjøre bruk av domener, som feks i Internet. Til sist bør en slik adresse kunne bli representert ved et alias som er lett å huske, eks ifi.uio.no, og navneoppslag bør da kunne foretas effektivt.

- c) Hva ligger i begrepet 'ruting' ?

  Ruting er en funksjon som utføres av svitsjene (ruterne) i et datanett. I grove trekk innebærer denne funksjonen at svitsjen leser mottager-adressen i pakkehodene som ankommer, og bestemmer ut fra denne hvilken retning pakken skal videresendes (velger fysisk link). Det finnes både dynamiske og statiske strategier for å foreta dette valget.

4. Ytelse (performance)

- a) Forklar begrepene 'båndbredde' (bandwidth) og 'gjennomstrømning' (throughput)

  'Båndbredde' benyttes ulikt avhengig av om det er snakk om analoge eller digitale systemer. I det første tilfellet måles båndbredde i Hertz (Hz), eks har en analog telefonlinje ca 3kHz båndbredde. Når det gjelder digitale systemer, refererer båndbredde vanligvis kapasiteten til en link målt i bit per sekund (bps), også kalt bitrate, overføringshastighet, transmisjonsrate etc.

  'Gjennomstrømning' er den faktiske datamengden som kan overføres på en link pr sekund (bit/s). Teoretisk maksimal båndbredde er ofte uoppnåelig pga. pakketap, flere brukere, maskin med for lav eksekverings kapasitet i forhold til ønsket senderate , etc.

- b) Hvor oppstår forsinkelse (delay) når en pakke sendes mellom to endemaskiner?

  Forsinkelsen oppstår flere steder: Protokollstakken i avsendermaskinen; unngå kopiering av pakker mellom lagenes buffer, bruk pekere som flyttes. Propagasjonstid; overføringsmediet introduserer forsinkelse i form av tiden det tar elektroner å forflytte seg i en kobberledning / fotoner i en fiber. Prosesseringstid; tiden pakken holdes tilbake i mellomliggende noder mens hode analyseres og det ventes på ledig linje videre. Utsendingstid; avhengig av kapasitet på linken (eks 1Mbps linje, 10kb pakke gir 10ms utsendingstid for å dytte alle bit ut på linja. Retransmisjon av feilretting; av pakker ved feilsituasjoner, inkl. evt omruting av pakker ved nodefeil.

- c) Gjør rede for begrepet 'tjenestekvalitet' (QoS). Hvilke elementer kan inngå?

  Viktige parametre er: overføringskapasitet, ende-til-ende forsinkelse, variasjon i forsinkelsen (jitter), akseptabelt pakketap

5. Protokoller

- a) Gi en definisjon av begrepet 'protokoll' :

  En *protokoll* er regler for kommunikasjon mellom prosesser.

- b) Hva er hensikten med å benytte lagdelte protokoller (design og anvendelse) ?

  Lagdelingen er en måte å abstrahere systemet på, og gjør det lettere å organisere de oppgavene protokollstakken skal inneha. Hvert lag har sine dedikerte oppgaver, og tilsammen vil disse gjøre det mulig å tilby kompliserte tjenester. Fra et gitt lags perspektiv, er det kun laget direkte over og under som er av betydning: det brukes tjenester fra laget under (bygger på), og det tilbys tjenester til laget over. I tillegg må et lag kunne samarbeide/ kommunisere med tilsvarende lag på andre maskiner (piers). Lagdelte protokoller skal i prinsippet være modulære, dvs at man kan skifte ut en protokoll eller et lag i stakken med et annet uten endringer i omliggende lag, noe som krever veldefinerte grensesnitt mellom lagene. I virkelighetens implementasjoner gjøres det mange snarveier i protokollstakken for å oppnå bedre ytelse.

- c) Skisser OSI modellen og TCP/IP modellen. Hva består forskjellen mellom dem i? Hvorfor er de ulike?

  Se figur 1-20 og 1-21 i Tanenbaum Grensesnittet mellom lagene inkluderer såkalte SAP'er (Service Access Point), Siden et lag tilbyr tjenester til laget over, og disse tjenestene leveres via SAP, tegnes SAP inn i modellen på toppen av det tilhørende laget, gjerne også med en bufferkø med meldinger som skal leveres. Tjenestebruker er det laget som benytter tjenestene, og man kan gjerne sammenligne dette med prosedyrekall med tilhørende returverdier: feks vil transportlaget kalle en funksjon i nettverkslaget for å sende en pakke, og returen fra funksjonskallet er meldingene som kommer tilbake fra den kommuniserende motparten.

  Den største forskjellen ligger i at TCP/IP modellen ikke har egne lag for de funksjonene OSI modellen har skilt ut i Sesjons- og Presentasjonslag. Nettverkslaget er dessuten ikke formelt delt i tre lag, men IETF har spesifisert protokoller som behandler alle disse lagene. OSI er teoretisk og mer et rammeverk for diskusjon ang kommunikasjon i og med at den spesifiserer kravene til alle lag. OSI modellen er sjelden implementert i sin helhet, bl.a. pga ytelsesproblemer. De to modellene ble laget av to ulike organisasjoner (ISO og IETF/ARPA) med sine ulike syn på hvor mange lag et kommunikasjonssystem bør bestå av, og hva slags funksjonalitet hvert lag skal inneholde. Med tiden har modellene nærmet seg hverandre noe.

- d) Hvor i OSI modellen finner vi følgende element:

  tjenstekvalitet (QoS): brukes ofte om hvilken garanti et nettverk har for å sende pakker, i form av flytkontroll, retransmittering, båndbredde, forsinkelse / gjennomsnittlig forsinkelse. Ansvaret for dette ligger i de 4 nederste lagene av OSI modellen, med hovedvekt på transportlaget. Tjenestekvaliteten har en verdiøkning oppover i stakken (summerer tjenesterne).

samtaleunivers beskriver hvilke datastrukturer som skal utveksles mellom entitetene på applikasjonslaget samt hvilke operasjoner som skal utføres på disse dataene ved motta. Beskrivelsen er abstrakt, og blir først konkret når datastruktur og operasjoner lineariseres og kodes for utveksling mellom endesystemer. Nødvendig for kommuniserende systemer å ha en felles forståelse.

protokoll spesifiserer de regler som gjelder for kommunikasjonen mellom entiteter på samme lag i to systemer som kommuniserer med hverandre. Det kan finnes mange forskjellige implementasjoner av en og samme protokollspesifikasjon. Finnes på alle lag.

grensesnitt beskriver eksempelvis skillet mellom lag i modellen (tjenestegrensesnitt), eller skillet mellom to entiteter som kommuniserer på samme lag i ulike systemer (protokollgrensesnitt).

- e) Hva er forskjellen på en "De jure" og en "De facto" standard? Nevn eksempler på begge

  "De jure" betyr vedtatt, i denne sammenheng en standard bestemt av en autorisert, internasjonal standardiserings komité (ISO, ITU). "De facto" betyr at standarden har blitt til gjennom utbredt bruk, ofte spesifisert av industrikonsortia eller uformelle organisasjoner som IETF (og er mindre tidkrevende enn byråkratiets De jure, altså finnes det mange standarder av denne typen)

6. Litt av hvert

a) **En ulempe med noen kringkastingsnett er den kapasiteten som sløses bort når multiple vertsmaskiner prøver å aksessere kanalen samtidig. Anta et enkelt eksempel der tiden deles inn i diskrete "slots", der *n*vertsmaskiner prøver å benytte kanalen med sannsynlighet *p* i løpet av hvert slot. Hvilken brøkdel av slot´ene sløses bort pga. kollisjoner?**

Vi skiller mellom $n + 2$ hendelser. Hendelsene 1 til n innebærer at tilsvarende vertsmaskin med hell prøver å bruke kanalen, dvs. uten at kollisjon oppstår. Sannsynligheten for hver av disse hendelsene er $p(1-p)^{n-1}$. Hendelsen $n + 1$ er en ledig kanal, med sannsynlighet $(1-p)^n$. Hendelsen $n + 2$ er en kollisjon. Siden disse $n + 2$ hendelsene representerer alle mulighetene, må summen av deres sannsynligheter være 1. Sannsynligheten for en kollisjon, som er lik brøkdelen av slot´er som sløses bort, er da $1-np(1-p)^{n-1}-(1-p)^n$.

b) Når en fil overføres mellom to datamaskiner, har man to mulige kvitterings-strategier. I det første tilfellet, deles fila opp i pakker som kvitteres individuelt av mottaker; filoverføringen som sådan blir imidlertid ikke kvittert for. I det andre tilfellet, kvitteres ikke pakker individuelt, men hele fila kvitteres når den har ankommet. Diskutér disse to tilnærmingene.

Hvis nettet tenderer til å miste pakker, er det best å kvittere for hver enkelt, slik at mistede pakker kan retransmitteres. Hvis nettet derimot er meget pålitelig, vil det å sende kvittering på slutten av hele overføringen spare båndbredde i det normale tilfellet (men dette krever at hele fila retransmitteres selv om bare en eneste pakke mistes).

c) Et bilde er 1024 x 768 piksler med 3 bytes/piksel. Anta at bildet ikke er komprimert. Hvor lang tid tar det å overføre det over en 56 kbps modem-forbindelse?

Bildet er 1024*768*3 bytes eller 2.359.296 bytes. Dette er 18.874.368 bit. Ved en hastighet på 56.000 bit/s, tar det omtrent 337s å overføre bildet.

d) Trådløse nett er lette å installere, hvilket gjør dem billige fordi installasjon av nett vanligvis koster mer enn utstyret. Likevel har de også noen ulemper. Nevn to av dem.

En ulempe er sikkerheten. Enhver person, som tilfeldigvis er i bygningen, kan lytte på nettet. En annen ulempe er pålitelighet. Trådløse nett introduserer vanligvis mange feil. Et tredje potensielt problem er levetiden for batterier, siden de fleste trådløse innretninger er mobile.

e) Nevn to fordeler og to ulemper ved å ha internasjonale standarder for nettverks-protokoller.

En fordel er at hvis alle bruker standarden, kan alle snake med alle. En annen fordel er at utstrakt bruk av enhver standard vil gi brukerne av de resulterende produktene en økonomisk gevinst, slik som f.eks. er tilfellet med VLSI chip´er. En ulempe er at de politiske kompromissene som må gjøres for å komme fram til internasjonale standarder, ofte leder til teknisk dårlige standarder. En annen ulempe er at, når en standard er tatt i utstrakt bruk, er den vanskelig å endre, også selv om bedre teknikker eller metoder oppdages. Dessuten kan den allerede være foreldet når den blir vedtatt.

f) Hva er forskjellen, hvis noen, mellom demodulator delen av et modem og koder delen av en codec?

En koder mottar et vilkårlig analogt signal og genererer et digitalt signal fra det. En demodulator mottar en modulert sinus-bølge og genererer et digitalt signal.

g) PPP er mye basert på HDLC, som benytter bit stuffing for å unngå at tilfeldige flag bytes inne i datadelen skal skape forvirring. Gi minst en årsak til at PPP bruker byte stuffing isteden.

PPP ble designet for å bli implementert i programvare, ikke i maskinvare slik som HDLC nesten alltid er (nettverkskort). I en programvare-implementasjon er det mye enklere å jobbe med bytes enn med individuelle bit. I tillegg ble PPP opprinnelig designet for å bli brukt sammen med modemer, og disse mottar og overfører data i enheter på 1 byte ikke 1 bit.

h) Hva er det minste overhead´en ved å sende en IP-pakke når PPP brukes? Her skal du bare regne med overhead´en som introduseres av PPP selv, og ikke IP-hode overhead.

Det minste en ramme kan ha, er to flag-bytes, en protokoll-byte og to sjekksum-bytes, altså totalt fem overhead bytes pr. ramme.

1

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 **UDPporter**.

IP protokollen inneholder et felt i IP-hodet kalt ”**Protocol**”. Det peker ut protokollentiteten 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.

2

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.
3
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 tilstandsmaskin modeller for de to sidene!
Vi har alt nevnt de parametrene som må inngå i Connect-kallet. På klientsiden vil Connectkallet
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
Internettet
Modem &
telelinje
ISP PC
4
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 ISPmaskinen 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. PC
ISP maskinen kan tenkes å ha følgende struktur
Ruter-funksjonen vil, via en fremsendingstabell, holde greie på hvilket grensesnitt IPadressen 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
Link
Fysisk
IP DHCP
Applikasjon
Transport
Link
Fysisk
Ruter-funk. DHCP
Link
Fysisk
Internett PC
5
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)

6

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 akuttmottakes
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
7
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

1

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.
LAN
R R
LAN
A B
X.25
2

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 IPadresse.
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). Tjenesterepertoaret
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.
TCP UDP
IP
Ethernet
Applik.
Endemaskin
Adapsjon
Nett (PLP)
HDLC
Ethernet Fysisk
Ruter
IP
Ethernet X.25
3
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 nettog 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 IPpakken 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 nettadresser.

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

Dette er det vel svart på allerede.

4

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?

L
A
N
LAN
LAN
T
R_2
R_1
5

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 "Hiddenterminal"

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

LAN A LAN C LAN D
LAN B
LAN E
1001110101
Manchester
6

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}.
A B
Satellitt
7
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
$1 - n * p = 0.99$ 🕓 $n = (1 - 0.99)/p = 10^3$.
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.
8
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];
6. 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
9
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.

Remote Procedure Call (RPC)
stub stub
transp transp
net net
Client proc
import export
Directory(Name) Server
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
Client Server

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
11 7 0 3 1 1 0 6 7 4 5 9
10
tjener-maskinen eksporterer alle sine prosedyrer som det er mulig å kalle fra en klient til en navnetjener,

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 kodegenerator 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 minsten-
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).

Løsningsforslag, inf3190 Vår 2004, oppgave 1 til 4:

Oppgave 1.

Et ramme-orientert data-overføringssystem opererer med en datarate på 512kb/s med en rammelengde på 512 bytes over en langdistanse link som produserer en forplantnings-forsinkelse (eng: "propagation delay") på 20ms. Et flytkontroll system som benytter en vindusmekanisme er nødvendig. Bestem den minste vindusstørrelsen som sikrer optimal gjennomstrømning (eng: "throughput").

Løsning:

Optimal gjennomstrømning oppnås dersom følgende relasjon gjelder:

$Nt_f > 2t_d + t_f \Rightarrow t_f = 2t_d/t_f + 1$ (antar at overføringstiden for ACK er neglisjerbar)

der N er vindusstørrelsen, t_f er rammeoverføringstiden og t_d er forplantningsforsinkelsen.

Forplantningsforsinkelsen er 20ms, og rammeoverføringstiden er:

t_f = rammelengde/bitraten = (512*8)/512000 = 8ms

Ved innsetting fås følgende relasjon: N > (2*20)/8 + 1 = 6

Minimum vindusstørrelse for effektiv utnyttelse av linken er således 6, og en vindusstørrelse på 7 burde derfor være adekvat. Dette krever 3 bit for å spesifisere en ramme innen vinduet til enhver tid.

Oppgave 2:

En punkt-til-punkt satellitt overførings-link som knytter sammen to datamaskiner benytter en stop-and-wait ARQ strategi med følgende karakteristika:

Data-overføringsrate = 64kb/s

Ramme-størrelse, n = 2048 bytes

Informasjons-bytes pr. ramme, k = 2043 bytes

Forplantning-forsinkelse (eng: "propagation delay"), t_d = 180ms

Størrelsen på kvitteringsmeldinger, a = 10 bytes

Round trip prosesseringsforsinkelse, t_p = 25ms

Bestem gjennomstrømningen (eng: "throughput") og link utnyttelsen.

Løsning:

Rammeoverføringstiden, t_f, er: t_f = (2048*8)/6400 = 0.256 s

Overføringstiden for kvitteringsmeldingen, t_a, er: t_a = (10*8)/64000 = 1.25 ms

Total tid for å overføre rammen og motta en kvittering er:

$t_f + t_a + t_p + 2t_d$ = 0.256 + 0.0012 + 0.025 + 0.36 = 0.642 s

Gjennomstrømningen er da: k = (2043*8)/0.642 = 25.5 kbs

Link utnyttelsen blir da hvis vi ser bort ifra t_a og t_p: $U = t_f/(t_f + 2t_d)$ = 41.56%

Oppgave 3:

En syklisk kode har generatorpolynom $x^3 + x + 1$. Informasjonsbitene 1100 skal kodes og overføres.

Bestem:

i. det overførte kodeordet,

ii. resten som oppnås hos mottakeren hvis overføringen er feilfri (vis utregningen)

iii. resten som oppnås hos mottakeren hvis en feil opptrer i bit 4 (regnet fra venstre)

Løsning:

i.

Generatorpolynomet, $G(x)=x^3 + x + 1$ uttrykt som et binært tall er 1011.

Først adderes tre nuller til informasjonsbitene for å produsere F(x)=1100000.

Divisjon av F(x) med generator-polynomet G(x) ved bruk av modulo-2 divisjon gir:

1100000 : 1011 = 1110

1011 __

1110
1011__
1010
1011_
010 (rest)
Resten adderes så til informasjonsbitene istedenfor nullene for å gi det overførte kodeordet 1100010.
ii.
Hvis overføringen er feilfri, dividerer vi de mottatte data med G(x), og får:
1100010 : 1011 = 1110
1011___
1110
1011__
1011
1011_
0000 (ingen rest)
iii.
Hvis en feil opptrer i bit 4, vil det resulterende bitmønstret være 1101010, hvilket resulterer i:
1101010 : 1011 = 1111
1011___
1100
1011__
1111
1011_
1000
1011_
011 (rest)
Oppgave 4:
En melding som består av 2400 bit skal sendes over et internett. Meldingen leveres til Transportlaget som hekter på et 150-bits hode. Deretter leveres meldingen til Nettlaget som benytter et 120-bits hode. Nettlags-pakker overføres via to nettverk som begge benytter et 26-bits hode. Destinasjons-nettverket aksepterer kun pakker som er opptil 900 bit lange. Hvor mange bit, inklusive hodene, blir levert til destinasjons-nettverket?
Løsning:
Denne oppgaven kan tolkes på følgende to måter, som begge bør aksepteres:
Alternativ 1: Hodet på 26 bit som brukes i det første nettet strippes av i ruteren mellom de to intermediære nettene. Nytt 26 bits hode legges på for destinasjonsnettet. Dette gir følgende pakke-innhold i begge de intermediære nettene:
26 120 754
Alternativ 2: Pakkene som sendes gjennom det første nettet innkapsler pakker med hode for destinasjonsnettet. Dvs. 26 bits hode adderes til den innkapslete pakken i det første nettet og strippes bort i ruteren. Dette gir følgende pakker for det første intermediære nettet:
26 26 120 754
Data pluss transport-hode = 2400 + 150 = 2550 bit, dvs. at 2550 bit leveres til nettlaget.
Hode nett

1 og 2
Hode
nettlaget
Transport data
Hode for
nett 1
Hode
nettlaget
Hode for Transport data
nett 2

Siden destinasjonsnettet bare aksepterer pakker på 900 bit, må disse dataene sendes i flere internet pakker der størrelsen ikke overstiger 900 bit.
Data-feltet for hver internet-pakke er: 900 – 26 - 120 = 754 bit.
Transportlags dataene innkapsles derfor i fire internet pakker på følgende måte:
Pakke 1: 754 bit data (slik at det gjenstår å innkapsle 1796 bit)
Pakke 2: 754 bit data (slik at det gjenstår å innkapsle 1042 bit)
Pakke 3: 754 bit data (slik at det gjenstår å innkapsle 288 bit)
Pakke 4: 288 bit data
Pakkene 1, 2 og 3 er hver på 900 bit. Pakke 4 er på 288 + 26 + 120 bit, dvs. 434 bit lang.
For alternativ 1, leveres følgende bit til destinasjonsnettet: 3*900 + 434 = 3134 bit.
For alternativ 2, leveres antall bit for alternativ 1 pluss 4 hoder for nett 1, dvs.:
3134 + 4*26 = 3238 bit.