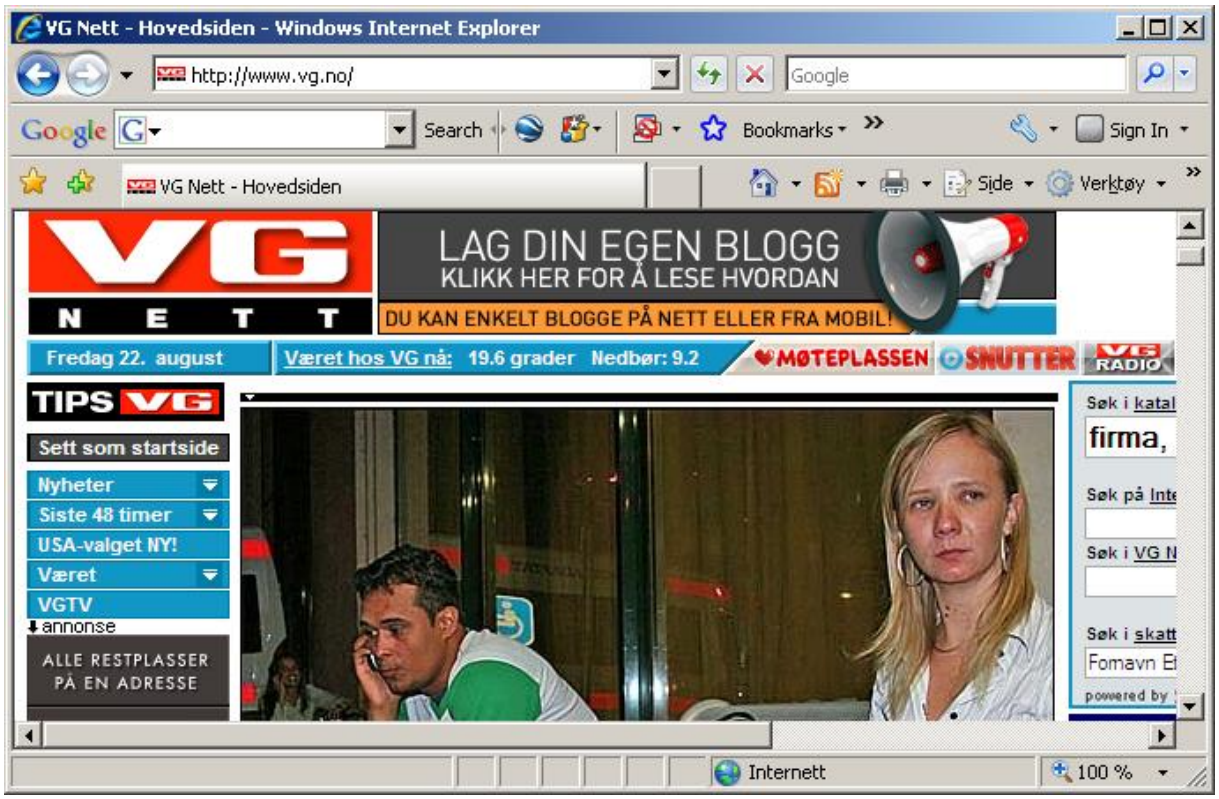
Kompendium i Internetteknologi 2011

Contents

Om Delemnet

Dette kurset er en innføring i virkemåten til Internett. For en webdesigner er dette nyttig kunnskap da det gjør det lettere for deg å lage effektiv design, feilsøke situasjoner som oppstår, vurdere ulike leverandører av Internett-tilkopling opp mot hverandre o.l. For studenter på andre programmer er det nyttig å ha grunnleggende kunnskap om feltet for å kunne kommunisere bedre med nettverksleverandører og systemansvarlige og for å bedre vite hva Internett faktisk er og dermed hva det kan levere.

Mer konkret så skal vi se på hva som skjer "under panseret" på Internett fra man legger inn en webadresse i nettleseren til siden vises. Kort fortalt og forenklet, så involverer nedlasting av hjemmesiden til VG:



1. Jeg skriver inn URLen http://www.vg.no. "http" angir her hvilken applikasjonslagsprotokoll som nettleseren skal bruke. "www.vg.no" er DNS-adressen til datamaskinen ("http-tjeneren") som nettsiden befinner seg på.
2. Browseren (Internett Explorer) spør operativsystemet (f.eks. Windows) om å få opprettet en socket ("TCP port"), og får et positivt svar. Så lager nettleseren din en http-melding som (forenklet) inneholder teksten:
   GET / HTTP/1.1
   (Dette betyr: få hovedsiden på serveren til VG.)

3. Samtidig sendes en DNS-forespørsel til din internettleverandørs DNS-tjener om hva IP-adressen som tilsvarer www.vg.no er. Når nettleseren har fått svaret så lager:
4. TCP/IP-delen av operativsystemet et TCP-segment som inneholder browseren sitt avsender-portnummer, og mottaker-port 80 (som er standard for http).
5. Operativsystemet lager et IP-datagram som inneholder din IP-adresse og VG sin IP-adresse.
6. Så lager operativsystemet (typisk) en Ethernet-ramme med MAC-adressen til ditt nettverkskort som avsenderadresse og sender denne til ruteren din.
7. Ruteren leser VGs IP-adresse, slår opp i en tabell, og videresender datapakken til neste ruter, som igjen slår opp i en tabell og videresender osv. Fra min PC (Oslo sentrum) til VG sin vev-tjener (Oslo sentrum) var datapakken min innom 15 forskjellige rutere på veien.
8. Nå ville du kanskje tro at VG sin http-tjener svarte med å sende deg VG sin hjemmeside. Det gjør den ikke. Først etter at operativsystemet ditt har fått en kvitteringsmelding om at http-tjeneren finnes og lytter, så sender den først sin egen kvitteringsmelding på at så er oppfattet. Først nå sendes faktisk HTTP-meldingen: GET / HTTP/1.1
9. Nå svarer http-tjeneren med å sende en HTTP-melding som innholder koden 200 OK, en del informasjon om serveren, og til slutt begynnelsen på web-siden du oppga URLen til i pkt. 1. Der etter lastes bilder og andre elementer ned.

Dersom du ikke forsto veldig mye av denne forklaringen så er ikke det noe problem. Målet med dette kurset er nettopp at du skal tilegne deg kunnskapene og det tekniske begrepsapparatet som er nødvendig for å forstå hva som foregår "under panseret" på Internett.

Det foregår en god del ting utover det som er beskrevet over. Her var meningen kun å gi en liten forsmak. I hvert av punktene over er det mange ting som kan gå feil, og mulige flaskehalser som må passeres; men alle feil og forsinkelser vil for sluttbrukeren se likedan ut: websiden de har bedt om kommer ikke opp.

Det vi skal lære i dette kurset er hvordan infrastrukturen Internett er bygd opp og henger sammen. Dette involverer å lære hvordan reglene for datakommunikasjon (*protokollene*) som Internett bruker er bygd opp og brukes.

Målet for studieveiledningen er å vise hvordan de prinsippene og teoriene som er nøye beskrevet i pensum-boken er tatt i bruk på din egen maskin og din konsekvensene for din Internett-tilkopling. Det tas derfor i størst mulig grad utgangspunkt i praktiske forsøk og eksempler som du også kan følge på din egen maskin.

Generell datakommunikasjon og Applikasjonslaget

Mål

Læreplanens mål

Mål for opplæringen er at eleven skal kunne

- definere begrepene Internett, protokoll og header, samt forklare sammenhengen mellom header og protokoll
- forklare forskjellen på linje- og pakke-svitsjing. Diskutere fordeler og ulemper ved disse teknologialternativene.
- forstå og forklare årsaken til ende-til-ende-forsinkelsen gjennom nettet med utgangspunkt i begrepene: prosessering-, køing-, utsendelse- og overførings-forsinkelse.
- definere og diskutere tjenestemodellene klient/tjener vs P2P, samt forklare forskjellen på forbindelsesorienterte vs. forbindelsesløse tjenestemodeller
- beskrive, og forklare fordelene ved en funksjonsbasert lagdeling i en protokollstack.
- definere TCP/IP-modellen og forklare hvilke oppgaver som løses på de ulike nivåene.
- benytte en "pakkesniffer" (Wireshark) til å inspisere virkemåten og formatene til ulike protokoller i sanntid, samt tolke og relatere sniffede data til overordnede prinsipper og teknikker i emnet.

- definere og forklare begrepene port og socket.
- beskrive funksjonsmåten og anvendelsesområdene til HTTP protokollen, her under oppbyggingen av meldingene og typiske meldingsutvekslinger.
- forklare oppbyggingen av en URL.
- beskrive funksjonsmåten og anvendelsesområdene til SMTP, POP3 og IMAP-protokollene, her under oppbyggingen av meldingene og typiske meldingsutvekslinger for SMTP.
- beskrive funksjonsmåten og anvendelsesområdene til DNS protokollen, her under oppbyggingen av meldingene og typiske meldingsutvekslinger; teste ut virkemåten til protokollen ved hjelp av NSLOOKUP (eller dig)

Datanettverk og Internett

Det viktigste å få med seg fra dette kapitelet er definisjonen av en (nettverks-) protokoll, tanken bak pakke-svitsjing og statistisk multiplexing, hovedårsakene til ende-til-ende-forsinkelsen og TCP/IP-modellen.

Internett

Internett er en teknisk infrastruktur som kopler sammen ca en milliard datamaskiner (pr 1.1.2011) verden rundt.

Maskinvaren som utgjør Internett består av vanlige PCer med nettverkskort (stasjonærere og bærbare), mobiltelefoner, dedikerte server-maskiner, opplysningstavler mm som tilbyr *tjenestene* som formidles over Internett. Disse

befinner seg på nettets rand ("network edge"). Sammenkoplingen foregår gjennom spesialbygd utstyr for datakommunikasjon i form av svitsjer, rutere som er koplet sammen med kabler (kobber, fiber) eller over radio-linker ("trådløst"). Mange av disse befinner seg i nettverkskjernen ("network core"). (Se fig 1.1 i læreboken).

Merk: Internett er IKKE det samme som World Wide Web (WWW). Internett er *infrastrukturen* som gjør det mulig å laste ned websider, sende og lese epost, dele filer osv. WWW er en av mange applikasjoner som benytter Internett.

Internett skiller seg fra det tradisjonelle og digitale telefonnettverket ("POTS") ved at det er *pakkesvitsjet*. Det tradisjonelle telefonnettverket er også et slags datanettverk, men det er *linjesvitsjet*. I et linjesvitsjet nettverk så opprettes det en forbindelse med dedikert overføringskapasitet (*båndbredde*) før dataoverføringen starter opp. Du slår først nummeret, det opprettes en linje gjennom telefonsentralene, det ringer hos mottager og så starter kommunikasjonen. Den reserverte kapasiteten er "bortkastet" når ingen sier noe. I et pakkesvitsjet nettverk brytes datamengden som skal overføres opp i små pakker som hver enkelt utstyres med avsender- og mottager-adresse. Pakkene rutes uavhengig av hverandre gjennom svitsjer og rutere fra avsender til mottager ut fra adressen som er lagt til på datapakken. Hver pakke bruker da *hele* den tilgjenglige båndbredden i pakkesvitsjede nettverk, i motsetning til hva som er tilfelle i linjesvitsjede, der de får dedikert en del av den samlede båndbredden.

Internett er upålitelig. Dette er et valg som ble gjort da Internett ble designet og utformet. I motsetning til tradisjonelle telenettverk der vi har enkle og billige telefonapparater som koples sammen via kompliserte og dyre telefonsentraler, så valgt man i Internett å la endesystemene ta seg av mest mulig, mens nettverket mellom dem skulle gjøres enklest mulig. Dette medfører at Internett ikke leverer noen som helst form for garantier for at pakker kommer frem eller hvor lang tid ting tar. Internett er "best effort", og det er opp til endesystemene å oppdage at pakker går tapt o.l. problemer.

Protokoller

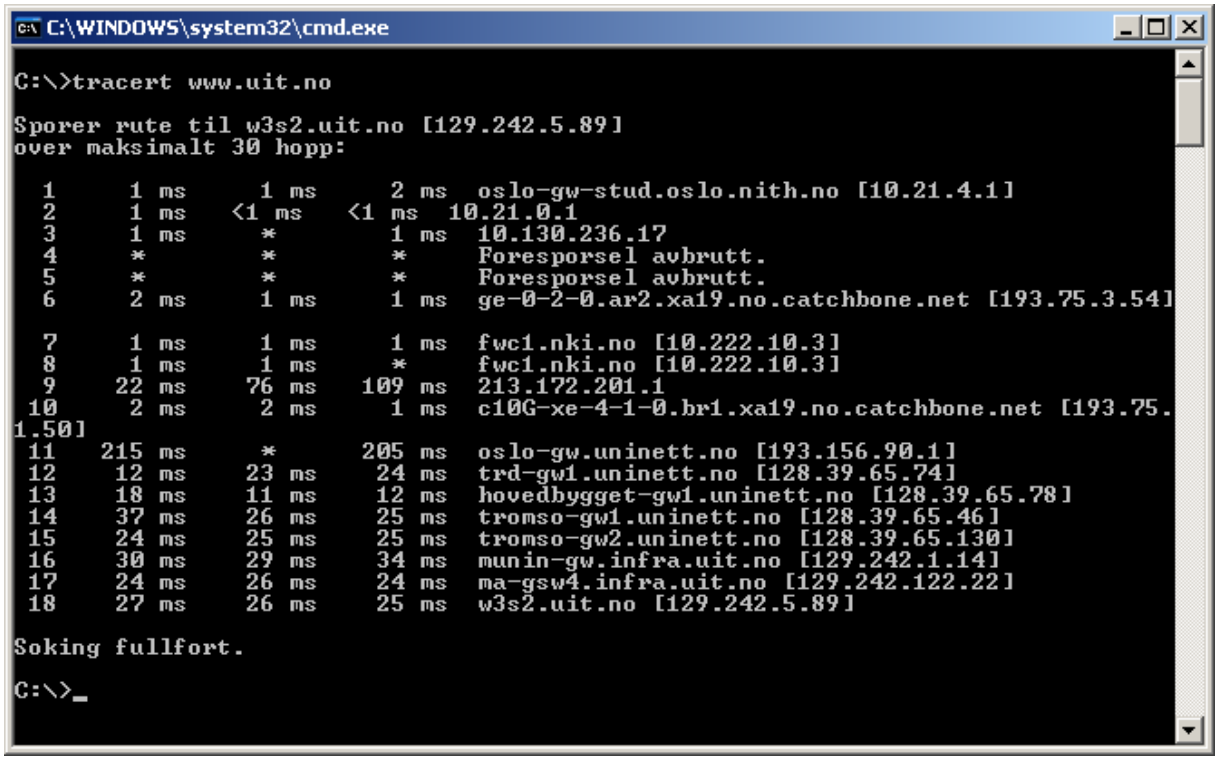
Internett er mer en noe annet definert gjennom de *protokollene* som benyttes. En protokoll bestemmer formatet og rekkefølgen på de meldingene som utveksles mellom to eller flere kommunikasjonspartnere. Protokollen bestemmer også hvilke handlinger meldingene skal utløse hos kommunikasjonspartnerne. En protokoll definerer dermed en syntaks ("hvilke verb og substantiv som benyttes, i hvilken rekkefølge"), en semantikk ("hva de skal bety") og angir timing for datakommunikasjon.

Internett-protokollene forvaltes av IETF (Internet Engineering Task Force) og er kjent som RFC'er (Request For Comment). Det finnes over 5000 RFCer, men ikke alle er like viktige. Det viktige er å forstå at protokollene er de *reglene* som gjør datakommunikasjon mulig. For eksempel så angir HTTP-protokollen hvordan browseren ("nettleseren") skal utforme en forespørsel om å få tilsendt en web-side og hvordan web-tjeneren skal svare. TCP-protokollen angir på sin side hvordan vi skal få til en pålitelig overføring over et upålitelig pakkesvitsjet nettverk. IP-protokollen bestemmer hvordan pakkene skal videresendes mellom routere. IEEE 802.11n m.fl. beskriver hvordan pakkene som overføres som radiosignal mellom en bærbar maskin og trådløs router skal se ut og behandles. En stor del av dette kurset handler om å studere protokoller.

De fleste protokoller definerer en eller flere *headere* som bærer den informasjonen som trengs for å benytte protokollen til å be om, eller utføre, tjenester. Å forstå virkemåten til en protokoll handler dermed i stor grad om å inspisere og forstå disse headerene.

Routing i Internett

For å få illustrert hvordan pakker flyttes gjennom Internett kan man bruke konsoll-applikasjonen tracert i et Windows kommandovindu. Under ser vi hvilken vei en pakke følger fra NITH i Oslo til webserveren til Universitetet i Tromsø:



Figur 1: Tracert NITH - UIT

I trinn 1 forlater pakken NITH, trinn 2 til 10 er innenfor nettverket til NITHs ISP (Internet Service Provider). Trinn 11 er ut av Oslo via Universitetsnettverket i Norge: UNINETT. Trinn 12 og 13 er mellomlanding i Trondheim. Trinn 14 til 18 er veien pakken følger i Tromsø frem til UiT sin webtjener.

Innenfor lokalnettverk (LAN – Local Area Network) er det mest brukte overføringsmediet for nettverkskommunikasjon en type kobberkabel som er kjent som UTP (Uskjermet tvunnet TrådPar). Disse finnes i ulike kategorier ut fra kvaliteten på kabelen. Kategori (Cat) 5, 6 og 7 er de mest vanlige i våre dager. For hjemmenettverk kan disse brukes som tilkopling til egen svitsj/router/modem, mens kabel ut fra hjemmet til ISP typisk enten er en DSL (Digital Subscriber Line), som også er kobberkabel, eller koaksialkabel for de hvis ISP også er kabelTV-leverandør. I kjernen av nettverket er det typisk fiberoptisk kabel som benyttes. Trådløse nettverk benytter radiosignaler, men på svært forskjellige frekvenser og med ulike båndbredder (Bluetooth, Wi-Fi, IEEE 802.11, satellitt, …)

Internetts kjerne (stamnettet)

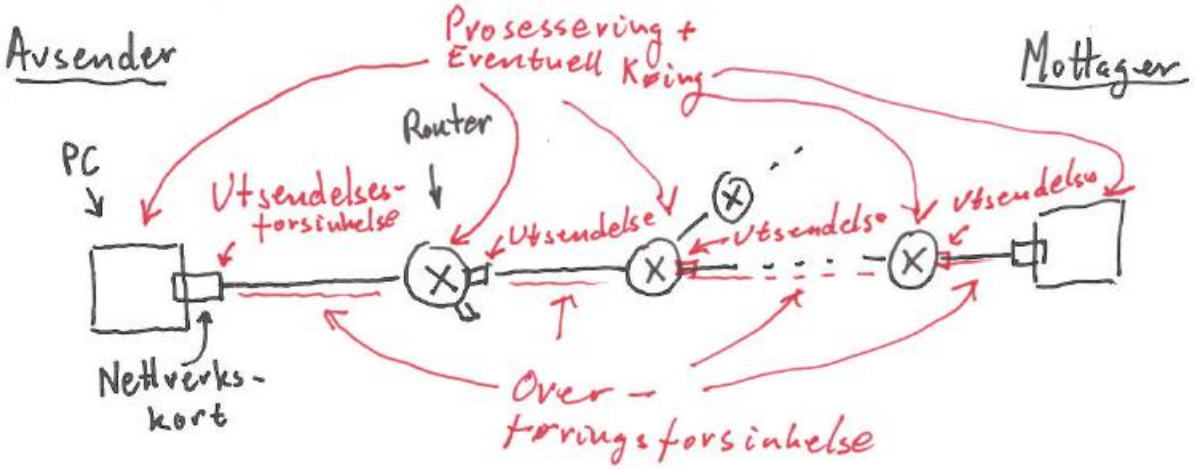
Internett er et *nettverk av nettverk*. Hvert av disse nettverkene er en ISP. Disse ISP-nettverkene varierer mye i størrelse og geografisk utbredelse. De største ring-1 (”tier-1”) nettverkene dekker og kopler sammen kontineneter (Europa – Amerika). På nivået under har vi ring-2 (regionale/nasjonale) nettverk slik som UNINETT i eksempelet over. På nivået under har man lokale ISPer som typisk er de som tilbyr Internett-tilkopling til privatkunder og bedrifter. ISPene på de ulike nivåene har avtaler seg i mellom om å utveksle pakke-trafikk. For å se et eksempel på hvordan en del av dette koples sammen i Norge kan du ta en titt på hjemmesiden til Norwegian Internet Exchange (http://www.uio.no/nix/) som i 2008/9 står for sammenkoplingen av om lag 20% av Internett i Norge. Resten foregår direkte mellom ISPene (”peering”).

Bitrate og repsonstid

Hvor lang tid det tar fra man ber om en tjeneste til den blir levert er avgjørende for brukeropplevelsen. I Internett er det flere faktorer som bestemmer denne responstiden. Datapakken som skal overføres er bygd opp av bit (0 eller 1, høy eller lav spenning, osv). Hvor mange bit du kan ”pumpe ut” på linja i et gitt tidsrom kalles egentlig for *bitrate*, men omtales oftest som *båndbredde*. Dersom bitraten er 10 Mbps, så kan du få 10.000.000 bit ut på 1 sekund. En typisk datapakke er på 1500 Byte, som er 1500 Byte · 8 bit/Byte = 12000 bit. Dersom denne sendes ut med bitrate 10 Mbps tar det $\frac{12000\ bit}{10000000^{bit}/_{sekund}} = 0{,}0012\ sekund = 1{,}2\ ms$. Dette kalles utendelsesforsinkelsen (”transmission delay”). Utsendelsesforsinkelsen er altså den tiden det tar å få data*pakken* ut i overføringsmediet. I et lokalnettverk er dette ofte den viktigste kilden til forsinkelse og mest avgjørende for responstiden. Selve signalet beveger seg med en hastighet på ca 2/3 av lyshastigheten i kobberkabel, dvs ca 200.000.000 m/s. Det tilsvarer at signalet går jorden rundt 5 ganger i løpet av et sekund! I løpet av den tiden det tar å sende ut pakken på en 10 Mbps linje ville det kunne ha forflyttet seg ca 24 km. Denne overføringsforsinkelsen (”propagation delay”) avhenger dermed kun av den geografiske avstanden mellom avsender og mottager. I et hjemmenettverk er det dermed typisk at den første biten i en pakke ankommer routeren lenge før den siste har forlatt nettverkskortet.

Utsendelsesforsinkelsen avhenger dermed av hvilken bitrate overføringsmediet støtter og vil derfor variere på hvert hopp langs ruten fra avsender til mottager. Overføringsforsinkelsen varierer mye mindre og er tilnærmet konstant pr meter.

På hver router (og svitsj) langsmed ruten vil pakkene måtte prosesseres for å finne mottageradressen og velge hvor de skal videresendes. Dette medfører for det første en prosesseringsforsinkelse: den tiden det tar å lese adressen og velge hvor den skal videresendes. Avhengig av hvor mye trafikk det er på den enkelte routeren på et gitt tidspunkt vil det også kunne oppstå en forsinkelse pga at pakkene må legges i kø. Køingsforsinkelsen vil øke med trafikken på den enkelte routeren. Dersom den samlede trafikken blir stor nok vil routeren begynne å droppe pakker, dvs den slutter å sende dem videre. Dette kalles pakketap.



Figur 2: Skisse av hvor forsinkelser oppstår

For å oppsummere så har vi da følgende årsaken til den samlede ende-til-ende-forsinkelsen:

- Utsendelsesforsinkelsen som er bestemt av bitraten til mediet.
- Overføringsforsinkelsen som avhenger av geografisk avstand
- Prosesseringsforsinkelse som avhenger av utstyret som prosesserer
- Køingsforsinkelse som avhenger av hvor mye trafikk det er på ethvert tidspunkt.

TCP/IP-modellen

Datakommunikasjon handler som vi forstår om å løse mange forskjelligartede oppgaver: alt fra å fortelle web-tjeneren hvilken side vi ønsker å se på til å bestemme hvilket spenningsnivå som skal kode et 1-tall på linjen mellom to nettverkskort. Internett er bygd ut fra en lagdelt arkitektur der vi løser de ulike typene oppgaver på ulike nivåer. Videre er tanken at underliggende lag yter tjenester som overliggende lag har behov for. Internett er bygd opp i henhold til TCP/IP-modellen som er oppsummert i tabellen under:

Navn på laget	Betegnelse på overføringsenhet	Viktigste oppgaver/funksjoner Exempel på protokoller/standarder
Applikasjonslaget	Melding (Message)	Støtte nettverksapplikasjoner Ex: HTTP, DNS, FTP, SMTP, POP3….
Transportlaget	Segment	Transport av applikasjonslagsmeldinger mellom klient- og tjener-sidene til en applikasjon; herunder mux/demux, ulike nivåer av pålitelighet med mer Ex: TCP, UDP,..
Nettverkslaget	Datagram	Routing av datagram fra/til vertsmaskin gjennom nettverkskjernen Ex: IP (v4 og v6), ICMP, RIP, OSPF, BGP,…
Datalinjelaget	Ramme (Frame)	(Pålitelig) Levering av ramme fra nabo-node til nabo-node Ex: Ethernet II, FDDI, IEEE 802.11 …
Fysisk	Bit	(Kode og) Flytte enkeltbit mellom kommunikasjonspartnere. Ex: 10BaseT, …

I applikasjonslaget så løser vi problemet med hvordan *yte en tjeneste* gjennom å definere en protokoll som angir hvordan man spør om den og hvordan den leveres. På transportlagsnivå løses problemet med hvordan vi skal sikre at applikasjonslagsmeldingene blir levert til riktig applikasjon (portnummer), samt hvordan, og i hvilken grad, dette skal foregå på en pålitelig måte. Disse to lagene handler utelukkende om endesystemer. På nettverkslaget løser vi problemet med hvordan finne veien og få flyttet en datapakke fra endesystem til endesystem gjennom nettverkskjernens system av routere trinn for trinn. På datalinjelaget løses problemet med hvordan få overført datapakkene mellom nabosystemer i nettet på en pålitelig måte. På fysisk nivå løses problemet hvordan kode og overføre hver enkelt bit gjennom et overføringsmedium. De tre nederste lagene angår dermed også hvordan routerne som utgjør nettverkskjernen fungerer.

TCP/IP-modellen er det logiske kartet av funksjonsmåten til Internett og må kunnes utenatt. Den gir en grei måte å organisere egen kunnskap på, og er uunnværlig i all feilsøking.

Kontrollspørsmål til kapittel 1

1. Definer begrepet protokoll.
2. Forklar de viktigste forskjellene på pakke- og linje-svitsjing.
3. Definer bitrate.
4. Dersom alle sender data hele tiden med en konstant bitrate, er pakke- eller linjesvitsjing å foretrekke?

5. Hvor lang tid tar det å få overført en pakke på 1000 Byte over en 5000 km lang linje dersom bitraten er 1 Mbps?
6. Se på tracert-utskriften i Figur 1. Prøv å forklar de ulike forsinkelsene som fremkommer der med utgangspunkt i begrepene: utsendelse-, overførings-, prosesserings-, køings-forsinkelse og pakketap. Hva er for eksempel årsaken til at responstiden er lenger for trinn 11 enn trinn 12?
7. Forklar prinsippet bak en lagdelt protokoll-stack (”TCP/IP-modellen”).

Kapittel *2 – Applikasjonslaget*

Dette kapittelet forklarer hvordan noen av de mest brukte protokollene i applikasjonslaget fungerer. De viktigste protokollene i forhold til webdesign er HTTP og DNS.

Kommentarer til fagstoffet

Nettverksapplikasjoner er klin like all annen programvare i det at de yter en eller annen form for tjeneste. De skiller seg fra vanlige applikasjoner ved at de har flere bestanddeler som kjører på ulike endesystemer og som kommuniserer med hverandre over et nettverk. En web-applikasjon består for eksempel av nettleseren (”browseren”) som ber om side og web-serveren som kjører på serveren som oppbevarer siden. Web-serveren er (ideelt sett) alltid tilgjenglig via Internett, mens forespørslene kan komme når som helst fra hvem som helst. Dette kalles en *klient-tjener-arkitektur.* Alternativet til klient-tjener er en P2P-arkitektur (”Peer-to-peer” = ”likemann-til-likemann”) der alle endesystemer både kan tilby og be om tjenester. Vi skal se at applikasjoner slik som www og epost følger klient-tjener-modellen, mens applikasjoner som Bittorrent og Skype følger P2P-modellen.

For at delene i nettverksapplikasjonen skal kunne nå hverandre gjennom nettverket er de avhengig av å kunne angi hvilket endesystem (PC, server,…) som en melding skal sendes til. Det gjøres ved hjelp av en IP-adresse, noe vi studerer mer detaljert i den tredje delen av dette kurset der vi behandler *nettverkslaget*. I tillegg må de også vite hvilket program på endesystemet meldingen er til. Dette gjøres ved hjelp av *portnummer*. Denne typen adressering er en oppgave som løses på *transportlaget*, og er noe vi skal komme nærmere inn på i neste del av kurset. I første omgang holder det å vite at de to mest brukte transportlagsprotokollene er UDP og TCP. UDP er upålitelig i den forstand at man ikke vet om en pakke kommer fram eller ikke. TCP er pålitelig i den forstand at man får vite det når pakker ikke når fram, og har i tillegg mekanismer som sørger for at pakker da blir sendt på nytt.

Ulike applikasjoner skiller seg ikke bare ad ved at de følger ulike arkitekturer, de er også forskjellige mhp toleranse for datatap, minimumskrav til båndbredde, krav til responstid og krav til sikkerhet (kryptering). (Se figur 2.4 i læreboken) F.eks. så krever tjenesten epost at ingen data går tapt, men har nærmest ikke noe krav til båndbredde. Video-streaming på den annen side krever en minimumsbåndbredde, men har en viss tapstoleranse da ingen vil merke at 0,1 s

med video ikke kommer fram. Epost benytter derfor TCP, mens streaming og andre typer multimedia-applikasjoner ofte bruker UDP.

Det finnes etter hvert tusenvis av applikasjonslagsprotokoller, og det dukker opp nye hver dag. Her vil vi se ganske nøye på HTTP fordi dette er den absolutt mest brukte. Så skal vi raskt se på filoverføringsprotokollen FTP fordi den illustrerer en annen tilnærming enn den vi finner i HTTP. Så skal vi kort se på protokollen som benyttes for å sende epost: SMTP, fordi denne også viser en annen måte å overrføre filer på. I den sammenhengen kommer vi også inn på MIME, som er standarden som tillater overføring av multimedia-innhold med HTTP og SMTP. Til slutt skal vi studere virkemåten til DNS. DNS er protokollen som gjør at brukere slipper å huske IP-adresser slik som 193.69.165.21 og heller kan skrive vg.no.

HTTP

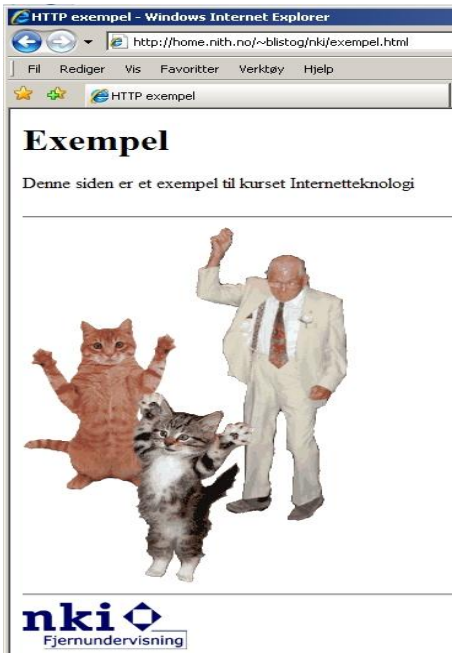
Vi kan se på en web-browser som et program som består av noen hoved-deler. I dette kurset skal vi ikke se på de delene som har med å tolke og vise frem HTML, CSS, Javascript o.l. Vi ser bare på det som har med nettverkskommunikasjon å gjøre. Vi kan da dele browseren i brukergrensesnittet (brukeragenten) som brukeren ser og samhandler med, og protokollmotoren som er den underliggende programvaren som omformer brukerens ønske om tilgang til en bestemt ressurs til en melding i det formatet som protokollen spesisfiserer og overlater denne forespørselen til operativsystemets TCP/IP-modul. Det er dermed operativsystemet som tar seg av nettverkskommunikasjonen, mens browseren bare formulerer forespørsler og mottar svar.

Når vi bruker en web-browser for å laste ned en web-side så ”bestiller” vi siden ved å angi en URL (”Uniform Resource Locator”). Et eksempel på en enkel URL er http://home.nith.no/~blistog/nki/url.html. Formatet tolkes av browseren din på følgende måte: Protokollen som skal brukes for nedlasting er http. Webserveren er home.nith.no. Det som skal hentes fra webserveren er ressursen med filnavn url.html som ligger i katalogen nki under hjemmeområdet til brukeren med brukernavn blistog.

Generelt så er formatet på en URL:
`skjema://brukernavn:passord@server:portnummer/filsti/filnavn?argumenter_til_server=argumenter#angivelse_av_passering_innenfor_side`,
der skjema er angivelse av protokoll (http, https, ftp, file e.l.), brukernavn og passord er mulig å oppgi, servernavn må oppgis, browseren legger automatisk til portnummer 80 dersom protokollen er http, serveren automatisk leter etter de mest brukte filnavnene dersom ikke er angitt, argumenter til serveren oftest går til ulike former for serverside-programvare, mens bokstavene etter en # viser til en bestemt plassering innenfor en side. (Det generelle formatet som URLen følger kalles URI og er definert i RFC 3986, se http://tools.ietf.org/html/rfc3986)

Den mest brukte HTTP-meldingen er GET-forspørselen. En typisk web-side består av en basis-side inneholder HTML-formatert tekst, samt referanser til andre

objekter (f.eks bilder, stilark o.l.) som skal vises innenfor siden (Se Figur 3 og Figur 4).



Figur 4: Eksempel på en webside

```
<html>
<head>
<title>HTTP exempel</title>
</head>
<body>
<h1>Exempel</h1>
<p>Denne siden er et exempel til
kurset Internetteknologi</p>
<hr />
<img src="hachacha9dn.gif"
alt="Et bilde">
<hr />
<img src="http://nettskolen.nki.no/pub/
design/fj_logo.gif"
alt="NKI fjernundervisningen">
</body>
</html>
```

Figur 3: HTML-koden med tekst og henvisninger til to bilder

For å laste ned siden sendes da en HTTP GET-melding av browseren (klienten). Denne meldingen kan se ut som følger:

```
GET /~blistog/nki/exempel.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Language: en-gb,no;q=0.5
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0)
Host: home.nith.no
Connection: Keep-Alive
```

Den første linjen sier da hvilken side vi vil hente samt at vi bruker versjon 1.1 av http-protokollen. De fire neste linjene forteller serveren noe om hva salgs browser det er som har sendt meldingen. Linjen ”Host” må være med i HTTP 1.1 meldinger fordi det er mulig å ha flere ulike webserver-*navn* på samme IP-adresse, og da må serverprogrammet kunne skille mellom disse. Linjen ”Connection: Keep-Alive” sier at serveren ikke skal kople ned TCP-forbindelsen umiddelbart etter at websiden er levert, men vente på flere forespørsler (persistent forbindelse).

Svaret fra serveren vil i dette eksempelet, noe forkortet, se ut som følger:

```
HTTP/1.1 200 OK
Date: Sun, 08 Feb 2009 12:41:15 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Sun, 08 Feb 2009 12:27:12 GMT
```

```
Accept-Ranges: bytes
Content-Length: 301
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
<head>
<title>HTTP exempel</title>
</head>
<body>
<h1>Exempel</h1>
<p>Denne siden er et exempel til…
```

Den øverste delen er da headeren, som består av to hoveddeler: Den første linjen er statusmeldingen (200 OK) som betyr at serveren har funnet siden og oversender den. Den andre delen av headeren består av info om serveren og web-siden som overføres. Om serveren får vi vite hva salgs type det er (Apache) og tidspunktet. Resten fortekker at siden er på 301 Byte, kodet som HTML med tegnsettet UTF-8. Til slutt følger selve dataene (HTML-siden vi ba om).

I eksempelet over vil så browseren lese og formatere HTML-siden, oppdage de to referansene til bilder (<img src="…) og sende nye GET-meldinger for å hente disse ned før de vises frem i browser-vinduet.

Den typiske meldingsutvekslingen er altså en GET-melding fra klienten (browseren) og en svarmelding fra serveren i form av statuskoden 200 OK fulgt av en header med "data om dataene" (meta-data) og ressursen vi har bedt om.

Dersom serveren får en forspørsel om en ressurs som ikke finnes så svarer den med statuskode 404 ("Not found"). Dersom siden vi ber om er passordbeskyttet kan den svare med statuskode 403 og browseren vil be om brukernavn og passord i et eget sprettopp-vindu. Systemet for statuskoder er bygd opp ut fra 5 ulike grupper av situasjoner som kan oppstå:

Koder	**Type**	**Kommentar**
1xx	Informativ	Lite brukt, til testing
2xx	Suksess	200 OK er mest brukt, men har også koder for opplasting av filer, behandling av argumenter fra URL o.l.
3xx	Omdirigering	Det typiske er melding om at side har blitt flyttet (301 eller 302)
4xx	Klient-feil	Klienten har sendt en forspørsel som enten ikke kan oppfylles, inneholder feil i formatet, eller krever ekstrainformasjon
5xx	Tjener-feil	Forespørselen ber om noe serveren ikke er i stand til å levere.

I tillegg til GET tilbyr HTTP/1.1 også metodene HEAD, POST, PUT, DELETE, TRACE, OPTIONS og CONNECT. PUT og DELETE er for å laste opp og slette filer på server. TRACE og OPTIONS er mest for feilsøking og eksperimentering. HEAD ber bare om headeren for en gitt ressurs uten å laste ned selve ressursen, og er også mest anvendt til feilsøking. POST benyttes typisk til å laste opp data som er lagt inn i skjema på en web-side. CONNECT benyttes mest for å opprette og holde i gang krypterte forbindelser med http-serveren.

HTTP-protokollen er *tilstandsløs* (”stateless”). Dvs at hver enkelt HTTP-forespørsel behandles uavhengig av alle tidligere transaksjoner mellom klient og tjener. HTTP-serveren forholder seg *utelukkende* til det som står i hver enkelt HTTP-forespørsel som derfor må inneholde alt serveren trenger å vite for å levere tjenesten man ber om. Dette leder til et problem dersom f.eks. ønsker å ha en handlekurv på et web-sted. Da trenger man en mekanisme som kopler forespørslene opp mot en bestemt browser-sesjon. Den vanligste løsningen på dette problemet er å benytte ”cookies”. Mekanismen består i all enkelhet av at serveren når den mottar en GET-forespørsel svarer på vanlig måte, men legger til en linje i headeren som inneholder en tekst a la ”Set-cookie: 123456789”. I alle senere forespørsler vil browseren legge til en linje ser det står ”Cookie: 123456789”. Dermed kan server-applikasjonen for fremtiden benytte koden ”123456789” som et ID-nummer for denne bestemte bruker-sesjonen. Cookien lagres i en fil lokalt hos brukeren.

For å bedre ytelsen og dermed brukeropplevelsen mellomlagrer de fleste browsere web-sider og andre nedlastede ressurser. Dersom man ønsker å laste ned siden på nytt kan browseren legge til en linje med ”If-modified-since:” i HTTP-headeren. Serveren vil da sjekke om ressursen har blitt oppdatert i forhold til den datoen som er angitt for den siden/ressursen vi har mellomlagret lokalt og bare sende tilbake en statusmelding type ”304 Not Modified” dersom ressursen ikke har blitt oppdatert/endret siden sist vi lastet den ned. Mange lokalnett og ISPer setter også opp proxy-servere for HTTP som benytter et tilsvarende mellomlager for alle brukerne de betjener. De enkelte browserene må da være satt opp til å sende sine GET-meldinger til proxy-serveren som deretter sjekker om den har en oppdatert versjon av siden/ressursen mellomlagret ved hjelp av en ”If-modified-since:” i GET-meldingen. Poenget er da at mye etterspurte sider blir mellomlagret på proxy-tjeneren mye nærmere brukerne og de opplever en kortere responstid og høyere bitrate enn de ville ha hatt hver for seg.

FTP

FTP er eldre filoverføringsprotokoll som i motsetning til http *er* tilstandsorientert. FTP startes vanligvis opp ved at man åpner en kontroll-forbindelse mot FTP-tjenerens TCP-port 21. Så må man først logge seg på med brukernavn og passord. Der etter kan man fra bruker-agenten benytte ulike kommandoer for å få listet opp filer, bytte mellom kataloger og laste opp og ned filer. Kontrollforbindelsen mellom serveren og klienten holdes oppe under hele sesjonen, mens *serveren* åpner en ny forbindelse fra port 20 hver gang det skal

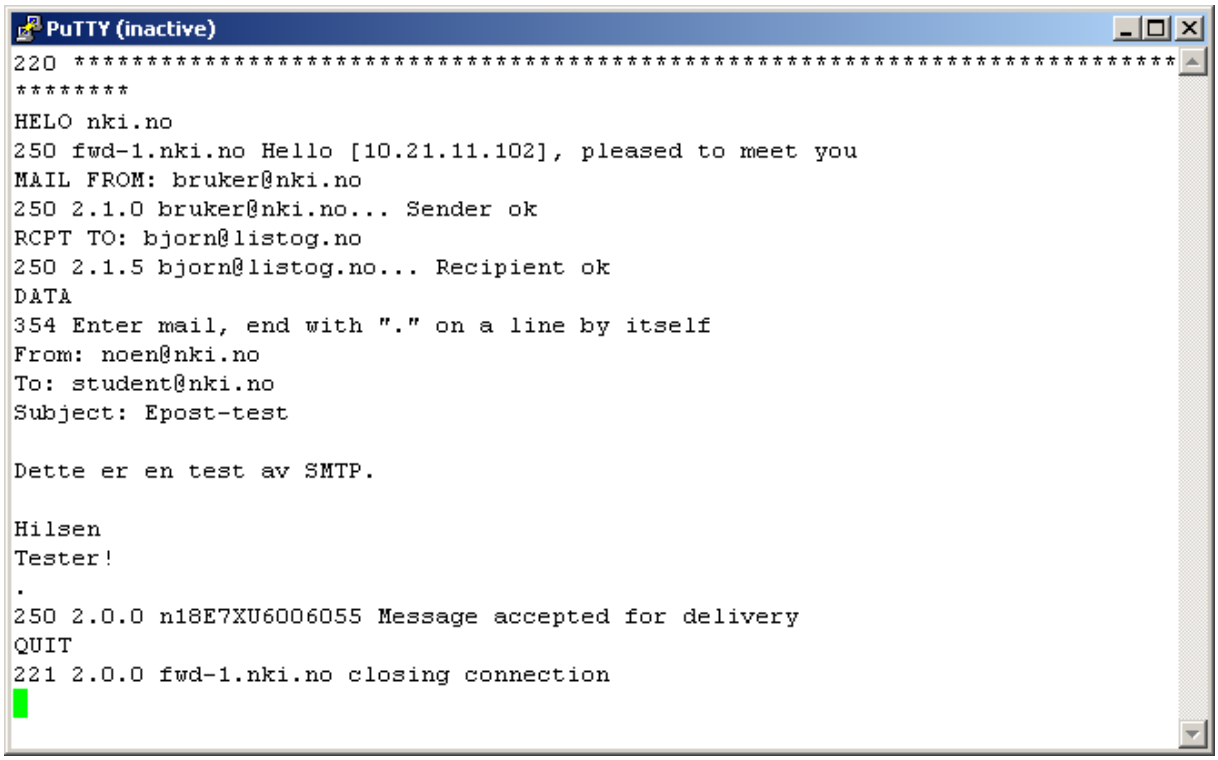
overføres data. Av sikkerhetshensyn tillater de fleste klienter og nettverks brannmurer ikke at serveren åpner en forbindelse, og man må derfor benytte det som kalles ”passive mode” der det er klienten som initierer forbindelsen mot port 20 på serveren. (Se læreboken s 116 for en nærmere beskrivelse av kommandoene og de tilhørende statuskodene i en FTP-sesjon)

FTP sender brukernavn og passord ukryptert, noe som ikke lenger oppfattes som sikkert nok. FTP er derfor i gang med å bli erstattet av sikrere protokoller (f.eks. SFTP og SCP). Den brukes fremdels likevel en god del til filnedlasting i tilfeller der autentisering av enkeltbrukere ikke er nødvendig (anonym nedlasting).

Epost

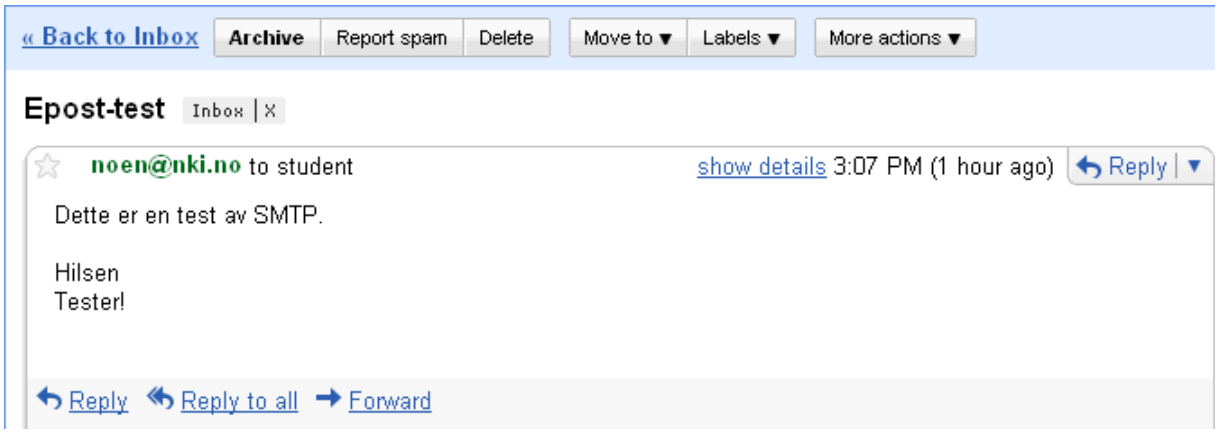
Mens HTTP og FTP tilbyr *synkrone* tjeneste, der vi er avhengig av en viss responstid og båndbredde for å oppleve at tjenesten blir levert, så tilbyr epost en *asynkron* tjeneste. Dette er akkurat som tradisjonelle brev der tjenesten er bygd på at det er opp til mottageren å svare nå det passer vedkommende. At epost er en asynkron tjeneste understrekes av det faktum at vi benytter vidt forskjellige protokoller til å sende epost og motta epost. Epost *sendes alltid* med SMTP (”Simple Mail Transfer Protocol”), men mottas enten med POP3, IMAP4, eller HTTP . (Se figur 2.17 i læreboken) Siden SMTP bare sender epost er den kjent som en *push*-protokoll, mens POP3 og IMAP4 er rene *pull*-protokoller.

I figuren under finner du en ”rå” sending av epost med SMTP. Bruker-agenten (epostprogrammet) starter med å opprette en TCP-forbindelse mot port 25 som er standardporten for SMTP-tjenere. Tjeneren svarer med statuskode 220. Brukeren starter med å sende en HELO-melding der man også angir hvilket DNS-domene avsenderen tilhører. Tjeneren svarer med en statuskode 250 OK. Så angir man avsenderadresse, og får statuskode i retur. Der etter angir man en eller flere mottager-adresser og får statuskoder i retur. Denne delen av eposten utgjør kommandoer til SMTP-tjeneren. DATA-kommandoen er det som tillater å skrive inn selve eposten. Denne består av to deler: en header og selve tekstmeldingen som er adskilt med to linjeskift. Meldingen sendes ved hjelp av kommando-sekvensen ”linjeskift-punktum-linjeskift”:



Figur 5: SMTP-sesjon

I figuren under ser du hvordan eposten ser ut på gmail.com. Legg merke til at det som her vises frem ikke er avsender og mottager vi oppgav til epost-tjeneren, men de adressene som ble lagt inn i headeren på tekstmeldingen! Alt det vi vanligvis får se i epostklienten vi benytter er data fra tekstfilen! SMTP-tjenere driver vanligvis ingen kvalitetsjekking av dette, noe som er en av årsakene til at det er så lett å lage "spam".



Figur 6: Eposten hos mottager

SMTP er definert i RFC 821 som skriver seg fra 1982. Det at protokollen er så pass gammel medfører at den har en del egenskaper som man har måttet kompansere for med ulike tilleggmekanismer. F.eks. tillater den kun bruk av det amerikanske tegnsettet (7 bit ASCII) i både headere og selve meldingene, og den har ingen innebygd mekanisme for vedlegg eller formatering av tekst. Både

norske tegn (æ,ø,å) og ulike typer vedlegg må derfor kodes om til ASCII-tegn før de overføres.

Norske tegn omkodes ofte ved å angi dem som Quoted Printable og legge inn hvilket tegnsett som brukes ved hjelp av spesielle koder. F.eks. så kan den norske teksten ”høre” i en epost bli omkodet til =?ISO-8859-1?Q?h=F8re?=, der F8 er den heksadesimale koden til tegnet ø i tegntabellen ISO-8859-1, som er det vanligst brukte vesteuropeiske tegnsettet.

Andre typer vedlegg slik som formatert tekst, HTML-filer, bilder, Word-filer, filmsnutter e.l. vil kodes om i henhold til MIME-standarden. Denne legger inn en ekstra header i ren ASCII-tekst der de to viktigste linjene angir hva slags type medie det er som overføres ved hjelp av linjen ”Content-Type:”, og hvordan det er kodet om til ASCII ved hjelp av linjen ”Content-Transfer-Encoding:”. Eksempelet under viser hvordan en bildefil av typen GIF kan bli overført som et vedlegg til en SMTP-melding:

```
MIME-Version: 1.0
Content-Type: image/gif; name="fj_logo.gif"
Content-Disposition: attachment; filename="fj_logo.gif"
Content-Transfer-Encoding: base64

R0lGODlhhgAzAMQAAPr7/NHT5G9zqomNuqKmyenq8gYHare61fPz+MTI3TE2hfDw9lhdnRAScJWZ
weLj7kZJkU1SliQofb3A2JygxX2Asjo+ihseeEFFjtrc6mBloq2wz/j4+wAAZQAAZv///yH5BAAA
///---Her har jeg fjernet mesteparten av den base64-omkodede bildefilen---///
AQKY97zoTa9618ve9rr3vfCNr3zZuwEBhAAAOw==
```

MIME tilbyr også et format for å angi at det er flere vedlegg.

For å få tilgang til epost benyttes oftest POP3, IMAP4 eller HTTP.
POP3 er den eldste av disse og gir enkel tilgang til brukerens postkasse på POP3-tjeneren, vanligvis gjennom TCP-port 110.
IMAP er en mer avansert protokoll som er designet med tanke på at brukere skal kunne oppbevare og organisere eposten sin på tjeneren i foldere.
HTTP benyttes til å bygge web-grensesnitt mot brukernes postbokser og omformatere innholdet til HTML-sider som overføres på vanlig måte til browseren.

DNS

DNS (”Domain Name System”) er katalog-tjenesten som gjør det praktisk gjennomførbart for de fleste brukere å bruke Internett. Den grunnleggende tjenesten som DNS-protokollen tilbyr er å oversette DNS-navn til IP-adresser. For å se på et enkelt eksempel kan man benytte verktøyet nslookup i et Windows kommandovindu (Se øvingsoppgave 2).

DNS er bygd opp som en ”distribuert database” av navnetjenere verden over. På toppen finnes det 13 rot-navnetjenere som i henhold til standarden har navnet ”.” (punktum), med tilnavn A.ROOT-SERVERS.NET til M.ROOT-SERVERS.NET. Disse har oppgaven med å holde oversikt over Top Level Domain navnetjenerne. TLDene holder oversikt over alle som har registrert et navn under et gitt toppnivådomene, for eksempel com, org, no, net, se, biz, kr, uk osv. Under dette

nivået finner vi et nivå av (stort sett) autoritative navnetjenere for organisasjoner og personer som har registrert domener, f.eks. nki.no, ibm.com, cia.gov, ietf.org o.l. Det er disse navnetjenerene som kan fortelle hva som er IP-adressen til f.eks. www.nki.no, nettskolen.nki.no fwd-1.nki.no o.l.

En DNS-adresse er hierarkisk bygd opp med toppnivået bakerst og den mest spesifikke delen først. nettskolen.nki.no er f.eks. et domenenavn som er registrert hos organisasjonen Nordid som forvalter no-adresser. For å få en .no-adresse må man ha et fortaksnummer registrert i Brønnøysund-registeret.

De fleste maskiner er satt opp med en DNS-klient som er satt opp til å sende *rekursive* DNS-meldinger til en lokal navnetjener. Rekursiv vil si at når jeg vil ha IP-adressen til nettskolen.nki.no så sender jeg en DNS-melding til den lokale navnetjeneren som er satt opp for mitt nettverk, og så foretar den alle oppslag som er nødvendige på mine vegne mot rot-, TLD- og autoritative navnetjenere. (Se figur 2.21 i læreboken.)

DNS-tjenere oppbevarer sone-filer som inneholder Resource Records (RRer). En RR er en linje som angir Navn, Type, Verdi og TTL (Time To Live), f.eks.

```
nki.no IN A 213.172.201.34 600s
```

I denne linjen så er nki.no domenenavn, IN A sier at typen er IP-adresse og at denne adressen er 213.172.201.34. TTL feltet sier at når en lokal navnetjener i ditt nettverk, eller på din maskin, mellomlagrer (”cacher”) denne adressen så skal den slettes etter 10 minutter. Andre viktige Typer er MX (SMTP-tjener)og NS (DNS-tjener). For eksempel så er den SMTP-tjeneren du benytter til å sende epost med helt avhengig av å kunne slå opp domene-navnet etter @ i adressen mot MX-feltet i navnetjeneren til mottageren.

Selv om DNS er en av de absolutt viktigste protokollene for at Internett skal fungere for brukerne benytter DNS den upålitelige transportlagsprotokollen UDP (standard port er 53). Årsaken til dette er at DNS-meldinger er små og får plass i ett segment. Kontroll med at vi får svar er lagt inn i den lokale DNS-klienten, som sørger for å sende forespørselen om igjen dersom det tar urimelig lang tid. Videre så krever TCP en oppkopling i forkant av dataoverføringen, noe som vi slipper med UDP og dermed normalt får raskere svar med UDP enn med TCP.

Kontrollspørsmål til kapittel 2

1. Lag en liste over minst fem forskjellige ikke-proprietære applikasjons-protokoller og forklar hvilke tjenester de tilbyr.
2. Hva er forskjellen på klient-tjener og peer-to-peer (P2P)?
3. Hvorfor benytte HTTP og SMTP TCP fremfor UDP?
4. Forklar hva en URL er, og hvordan den er bygd opp.
5. Forklar hvordan gangen i en vellykket HTTP-forespørsel er.
6. Forklar hvordan mellomlagring (”cacheing”) brukes sammen med HTTP og DNS. Hvilken fordel gir dette?

7. Lag en oversikt som viser hvilke trinn som inngår i sending og mottak av en epost.
8. Forklar hvorfor DNS er en så viktig applikasjon i Internett.

Studieenhet 2: *Transportlaget*

Mål for studieenhet 2

Læreplanens mål

Mål for opplæringen er at eleven skal kunne

- forklare hensikten med transportlagsprotokoller.
- forklare multiplexing/demultiplexing og sammenhengen med portnummer i TCP/IP-stacken
- forklare prinsippene bak pålitelig dataoverføring, her under mekanismene: kvitteringsmelding, omatt-sending, sekvensnummer, sjekksum, timer, pipelining, flyt- og trafikkork-kontroll
- forklare hvordan mekanismene over er tatt i bruk i TCP med utgangspunkt i feltene i TCP-headeren
- forklare oppbyggingen og anvendelsen av Internett-sjekksummen
- definere og forklare begrepene RTT, MSS og MTU, samt diskutere sammenhengen mellom disse
- diskutere bruksområder for UDP og TCP i sammenheng med tjeneste-typer og -krav
- Benytte netstat og andre vanlige verktøy til å inspisere og diagnostisere tilstanden på et gitt endesystem
- benytte en pakkesniffer til å inspisere ulike protokoller og formater i sanntid, samt tolke og relatere sniffede data til overordnede prinsipper og teknikker i emnet
- bruke NETSTAT til å inspisere TCP og UDP i bruk.

Kapittel *3 - Transportlaget*

Transportlaget løser oppgaven med å sørge for at programvare (applikasjoner) som kjører på ende-systemene kan adressere hverandre. Det sørger dermed for at applikasjonslagsprotokollene kan benyttes til å be om og levere tjenester. Vi skal begrense oss til å se på de to mest brukte transportlagsprotokollene: TCP og UDP.

Transportlaget i TCP/IP-modellen

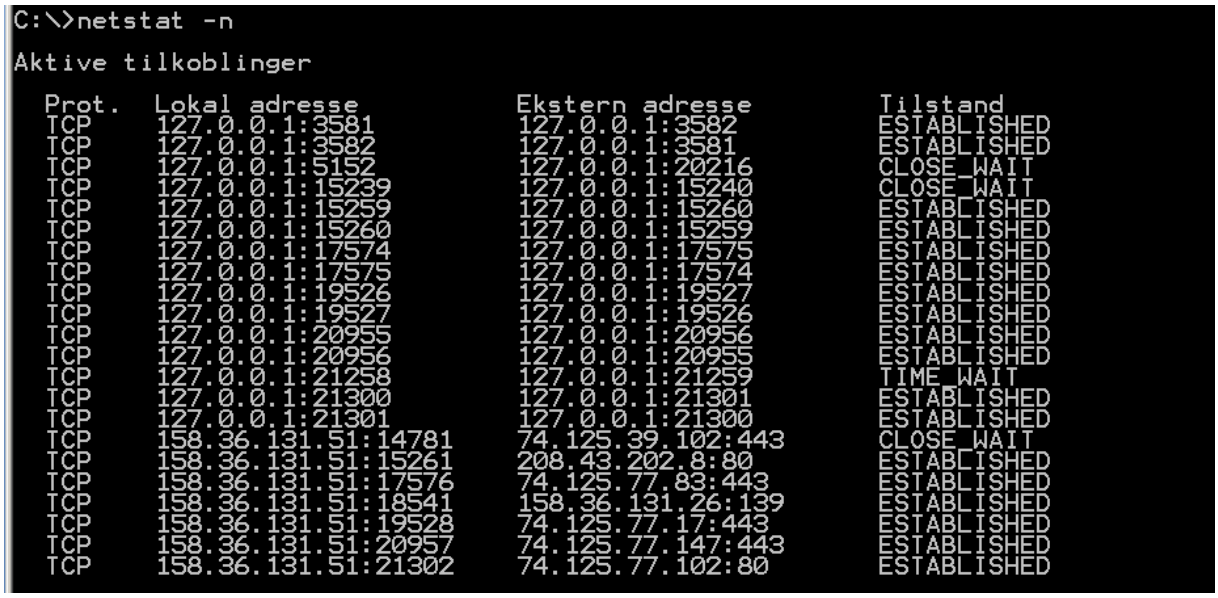
I TCP/IP-modellen så er oppgaven med å sørge for at prosesser (”kjørende programmer”) kan kontakte hverandre plassert på transportlaget. Dette kalles *logisk kommunikasjon* mellom prosesser. Adjektivet logisk brukes som motsetning til fysisk, og betyr at vi bare ser på hvilke regler som skal følges av kommunikasjonspartnerne. Vi ser ikke på hvordan resultatet faktisk og fysisk, trinn for trinn, oppnås.

Nettverkslaget i TCP/IP-modellen har med å opprette en logisk forbindelse mellom ende-systemer (”maskiner”) gjennom Internett. Dette gjøres ved hjelp IP-protokollen og diverse routing-protokoller. Disse garanterer ikke at datapakker kommer fram, bare at de blir forsøkt levert (”best effort”). Så mens nettverkslaget har med veien en datapakke følger gjennom Internett å gjøre, så har transportlaget med: 1) for både UDP og TCP sin del, hvordan vi sikrer at den blir levert til riktig program, og 2) i tilfellet TCP, med hvordan vi oppnår en pålitelig forbindelse over en upålitelig kanal.

Multipleksing/Demultipleksing: porter og portnummer

På datamaskinen din vil du ofte kjøre mange forskjellige programmer samtidig (”multitasking”). Flere av disse vil kunne være nettverksapplikasjoner og da må det finnes en måte å skille mellom dem på. Dette gjøres ved hjelp av portnummer. Portnummeret er ett (16 bits binær-) tall mellom 1 og 65535 som av TCP/IP-modulen i operativsystemet benyttes som adresse til en applikasjon (”et kjørende program”).

Ved å bruke standard-verktøyet *netstat* i et kommando-vindu kan vi få en oversikt over hvilke Internett-forbindelser vi har på gang:



Figur 7: netstat -n utskrift

I utskriften fra netstat-kommandoen får vi oppgitt til venstre hvilken transportlagsprotokoll det er som benyttes av forbindelsen. Så følger lokal og ekstern IP-adresse og portnummer. 158.36.131.51:14781 angir da at det lokale endepunktet for forbindelsen er på min maskin og til prosessen med portnummer 14781. Hvilken prosess dette er kan vi finne ved hjelp av kommandoen *netstat –nb*. Som vi ser av utsnittet under var dette GoogleCalendarSync.exe

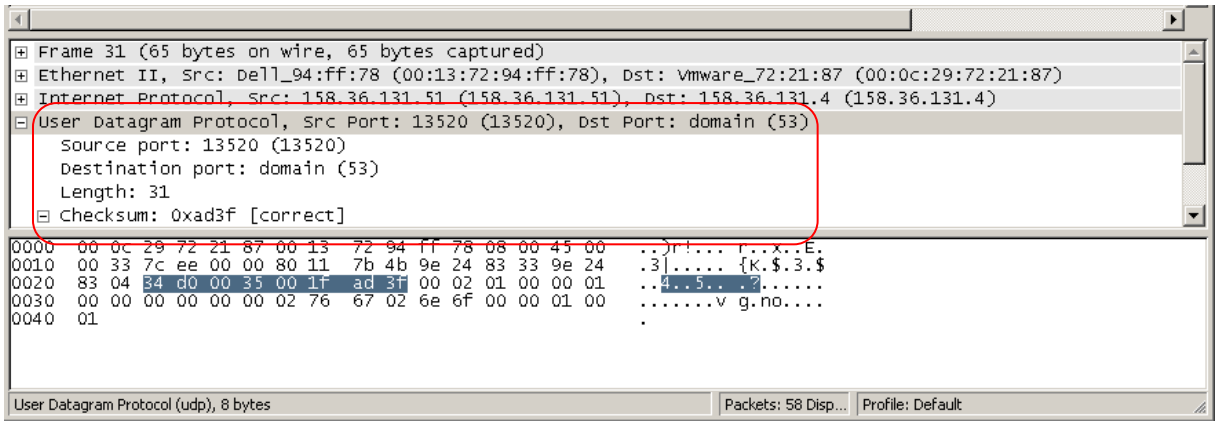
Figur 8: netstat -nb utskrift

Den eksterne adressen oppgis likeledes som ved parret IP-adresse:portnummer. Den siste kolonnen angir *tilstand* for forbindelsen. Vanligvis er denne etablert (ESTABLISHED), eller i gang med å koples ned (TIME_WAIT eller CLOSE_WAIT). I figuren kan vi se bort fra linjene med 127.0.0.1 da dette er IP-adressen for ”loopback”. Slike Internett-forbindelser med seg selv opprettes av ulike applikasjoner, oftest for å utføre ulike sikkerhetsfunksjoner.

Portnummer er delt opp i tre ulike klasser de velkjente (”Well Known Ports”), de registrerte (”Registered Ports”), og de dynamiske/private (”Dynamic and/or Private”). IANA fungerer som registrar foor portnummerne (se http://www.iana.org/assignments/port-numbers). De velkjente er portene fra 0-1023. Disse brukes på tjenermaskiner for mye brukte tjenester som http (port 80), smtp (port 25), ssh (port 22), https (port 443), osv. Disse portnummerene brukes kun av disse tjenestene, og vil aldri brukes som avsenderport av private brukere. De registrerte portene ligger mellom 1024 og 49151 og selv om det er registrert ulike tjenester som *kan* bruke disse som mottager-port, er de ikke like strengt regulert som de velkjente. Typisk så vil privatbrukere også kunne benytte disse tallene som avsenderport. De dymaniske portene er de fra 49152 til 65535 og disse er det ingen forhåndsdefinerte tjenester forbundet med og de kan benyttes fritt av både avsender og mottager.

Forbindelsesløs transport: UDP

Den enkleste av de to transportlagsprotokollene vi skal se på er UDP (”Unified Datagram Protocol”). UDP er ”lett”. Dersom vi ser på headeren så består denne av kun fire felter: avsender-port, mottager-port, lengde og en sjekksum.



Figur 9: UDP header for en DNS-spørring fra Wireshark

UDP-headeren er dermed på 8 byte = 64 bits; i motsetning til TCP-headeren som er på minimum 20 byte = 160 bits. Lengdefeltet i headeren angir den samlede størrelsen på UDP-segmentet, og har en maksimal verdi på 65535, hvilket betyr at det er plass til 65535 – 8 = 65527 byte med data (inkludert IP-headeren) i segmentet.

Ut fra feltene i headeren ser vi da at det UDP bidrar med er å identifisere avsender- og mottager-prosess (portnummerene), og gir mottager en mulighet for å sjekke at en mottatt datapakke er komplett og ikke har blitt endret i overføringen (lengde og sjekksum).

UDP er også kjent som ”Unreliable Datagram Protocol” fordi den er upålitelig. Protokollen sier ingen ting om hva som skal gjøres dersom en mottager oppdager at det er feil i lengde eller sjekksum, det er fullstendig opp til

programmet som benytter UDP. Siden finnes ett pålitelig alternativ i TCP er det grunn til å spørre seg: hvorfor bruker man UDP i det hele tatt?

I noen tilfeller vil man foretrekke UDP fremfor TCP fordi pålitelighet har sine kostnader. For noen typer tjenester så vil upålitelighet være en grei pris å betale for større hastighet og fleksibilitet. 1) TCP starter med å opprette en logisk forbindelse gjennom en ”three-way handshake”, UDP oppretter derimot ingen forbindelse. Fordelen med dette er at med UDP så sender du bare av gårde dataene, du har dermed ingen ventetid før du får jobben gjort. 2) Siden UDP ikke oppretter noen forbindelse mellom sender og mottager så kan den benyttes til å *kringkaste* data i et lokalnettverk (”broadcast”). Med TCP kan du bare nå en mottager om gangen, med UDP kan du sende til alle. 3)TCP mellomlagrer alle sendte data inntil du får bekreftet at de er mottatt; UDP mellomlagrer ikke, så du sparer minneplass og trenger heller ikke bruke like mye prosessorkapasitet på datakommunikasjonen. 4) TCP har en mekanisme som heter metningskontroll (”congestion control”) som medfører at en TCP-forbindelse bruker tid på å finne ut av hva som er den tilgjenglige overføringskapasiteten i nettverket og tilpasser seg denne hele tiden. Med UDP kan du sende så fort du klarer fra starten av og trenger ikke å ta noen hensyn til de andre som benytter nettverket.

For tjenester der høy overføringshastighet og lav responstid er viktigere enn at hver enkelt pakke kommer frem vil UDP ofte være å foretrekke. Dersom du skal overføre en fil vil du foretrekke TCP fordi de fleste filer er ubrukelige dersom de ikke er fullstendige. For f.eks. telefonsamtaler og streaming av film er høy overføringshastighet oftest viktigere enn at hvert sekund kommer fram, da vil UDP være å foretrekke.

Pålitelig data-overføring: prinsipper

Det fundamentale problemet som skal løses i alle former for dataoverføring er at kommunikasjonspartnerene ikke er telepatiske. Vi har allerede sett at vi definerer protokoller for å ha standardiserte beskjeder med fastlagt betydning som kan utveksles. Dette er en del av løsningen, men når vi kommer ned på transportlaget så er oppgaven også å sikre at applikasjonslag-meldingene kommer fram i samme stand som de ble sendt.

I kapittel 3.4-3.4.2 (s 214-225) bruker læreboken mye plass på å konstruere en transport-protokoll som de kaller RDT. I oversikten under er denne tilnærmingen oppsummert:
a) Problemet vi skal løse er hvordan oppnå **pålitelig** kommunikasjon mellom to forskjellige parter: sender og mottager over en upålitelig kanal.
b) Upålitelig kanal betyr at det finnes to problemer vi må håndtere: a) støy, b) tap.

Poenget med RDT er da å finne ut hvordan vi skal løse dette problemet. Vi tar det trinnvis.
1) På en perfekt kanal (støyfri og tapsfri) så må senderen kunne sende og

mottageren kunne motta. Det er det hele. Mer detaljert betyr det at når vi skal programmere protokollen vår så må sender-programmet ha en metode for å ta i mot meldinger fra applikasjonslaget og en metode for å sende dem videre. Tilsvarende for mottager. Dette er RDT 1.0
2) Så introduserer vi støy. Støy betyr i praksis at bits kan endre verdi på veien fra sender til mottager. 1 kan bli til 0, 0 kan bli til 1. Vi må ha en måte å **oppdage** at feil har oppstått på, og vi må ha en måte for mottager **å si fra** om at feil har oppstått på.
Oppdage feil kan vi gjøre ved hjelp av sjekksummer (se over).
Si fra om feil kan vi gjøre ved hjelp av kvitteringsmelding: ACK dersom sjekksummen stemmer, NAK dersom den ikke stemmer. NAK medfører da at meldingen skal sendes på nytt. Om att-sending er da sammen kvitteringsmeldinger den nye mekanismen. Dette er RDT 2.0
3) RDT 2.0 har alvorlige feil. Vi har ikke tatt hensyn til at når det er mulighet for støy på linja så kan den like gjerne ramme kvitteringsmeldingen som datameldingen. Det betyr at en melding kan ha kommet korrekt frem til mottager, men at kvitteringen tilbake til sender blir gal. Hvordan skal vi håndtere det?
Jo, vi lager en regel som sier at dersom en kvitteringsmelding inneholder feil så sender senderen forrige melding på nytt.
Det leder til et nytt problem. Vi må ha en måte å si fra til mottager at dette er samme melding som den allerede har mottatt. Dette løser vi ved å innføre sekvensnummer slik at mottager kan oppdage duplikat. Dette er RDT 2.1
3) For å "forenkle" protokollen vår fjerner vi NAK-meldingene og erstatter disse med vanlig kvitteringsmelding på sist korrekt mottatte melding. Det er RDT 2.2
4) Da gjenstår det å finne ut hvordan vi skal behandle den siste muligheten for feil, nemlig at pakker kan gå tapt.
Det gjør vi ved hjelp av en tidtaker / "timer" på hver melding. Regelen er at dersom vi ikke har mottatt kvitteringsmelding på en pakke innen rimelig tid, så anser vi den som tapt og sender den på nytt. Det er RDT 3.0

Mekanismene vi trenger for å oppnå pålitelig dataoverføring gjennom et upålitelig medium er dermed:

- sjekksum
- kvitteringsmelding
- sende om igjen
- sekvensnummer
- timer (tidtaker)

For å oppnå en akseptabel ytelse vi i tillegg muligheten til å sende flere segmenter før vi mottar kvittering. Dette for å kunne utnytte den tilgjenglige bitraten bedre. RDT slik den er beskrevet over en stop-and-wait protokoll. Vi sender ikke et nytt segment før vi har fått kvittering på forrige. I realistiske tilfeller medfører dette at vi ikke klarerer å fylle opp mer enn noen promille av den tilgjenglige båndbredden. Gitt at vi har en 100 Mbps linje Oslo-Nordkapp og

benytter datapakker på 1000 bit. Da tar det $\frac{1000\ bit/pakke}{100000000\ bit/sekund} = \frac{1}{100\ 000}\ sekund/pakke$ å få hele pakken ut av nettverkskortet og ut på linja. Avstanden Oslo-Nordkapp er ca 2000 km, og signalet beveger seg med ca 200000 km/s. Det tar dermed pakken $\frac{2000\ km}{200000\ km/s} = \frac{1}{100}\ sekund$å nå fram langsmed linja. Tiden fra vi sender pakken og til kvittering er mottatt (RTT, Round Trip Time) er dermed 20 millisekund + 0,001 millisekund. Med stop-and-wait klarer vi dermed maksimalt å utnytte $\frac{0{,}001\ ms}{20{,}001\ ms} = 0{,}00005 = 0{,}005\ \%$ av den tilgjenglige båndbredden. Det er ganske dårlig.

Løsningen er å åpne for at det kan sendes flere datapakker før man har mottatt kvittering. Med andre ord: å tillate at det er "flere pakker i lufta" samtidig. På engelsk kalles dette "pipelining". Vi må da lage noen tillegg til de mekanismene som er beskrevet over. For det første må vi velge hvor mange pakker som kan være sendt før vi får kvittering. Dette angir vi ved hjelp av en vindustørrelse ("window size") som i praksis sier hvor mange pakker vi kan ha sendt, og ikke mottatt kvittering på. For det andre må vi lage en mer komplisert mekanisme for hvor kvitteringsmeldinger.

Læreboken omtaler to forskjellige *prinsipper* for hvordan oppnå pipelining: Go-Back-N (GBN) og Selective Repeat (SR). I begge sender mottaker kvittering på pakkene innenfor sende-vinduet. Forskjellen ligger i *når* de kvitterer og *hvordan* de behandler feilsituasjoner:

- I GBN kvitterer mottaker kun for korrekt mottatte pakker i korrekt rekkefølge. Kvittering for et gitt sekvensnummer tolkes da av avsender som om alle pakker til og med den med dette nummeret er mottatt og OK. Dersom flere pakker har blitt sendt, men ikke kvittert for, blir de sendt på nytt. Bunnen av vinduet er dermed alltid det samme som sekvensnummeret til den siste pakken det har blitt kvittert for. Dette kan medføre at mange pakker som har blitt korrekt mottatt likevel blir sendt på nytt.
- I SR er poenget å kvittere for hver enkelt korrekt mottatte pakke. Avsender vil da kunne begrense seg til å sende på nytt pakker som har gått tapt eller har blitt mottatt med feil.

SR vil gi best ytelse, men er mer komplisert å gjennomføre. TCP har elementer av begge strategiene.

Forbindelsesorientert transport: TCP

TCP er svært forskjellig fra UDP i det at den er pålitelig og benytter alle mekanismene vi har gått gjennom over, samt noen flere andre i tillegg. TCP er antagelig den mest kompliserte protokollen av alle i TCP/IP-suiten.

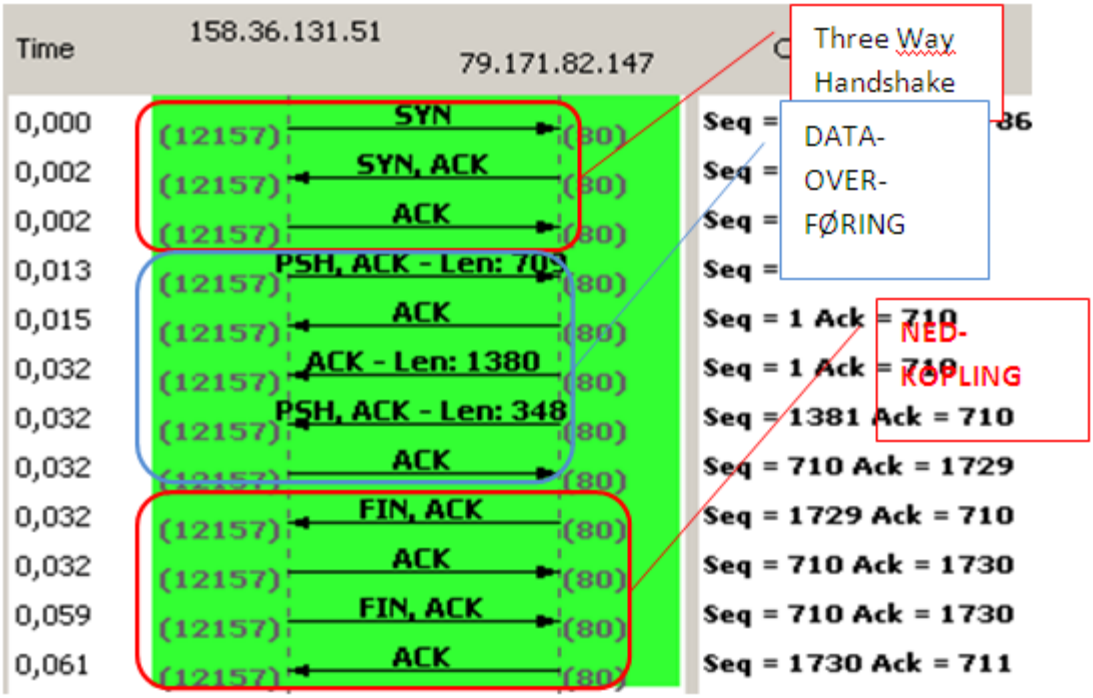
En TCP-forbindelse starter med en three-way handshake. Denne tjener flere formål. For det første får klienten bekreftet at tjeneren den ønsker kontakt med finnes. For det andre benyttes den til signalering av ulike verdier som kilent og

tjener må avtale før de begynner å kommunisere. I figuren under er det foretatt en Wireshark-fangst av nedlastingen av en liten web-side.

No.	Time	Source	Destination	Protocol	Info
9	*REF*	158.36.131.51	79.171.82.147	TCP	12157 > 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
10	0.002143	79.171.82.147	158.36.131.51	TCP	80 > 12157 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380
11	0.002194	158.36.131.51	79.171.82.147	TCP	12157 > 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
12	0.012895	158.36.131.51	79.171.82.147	HTTP	GET /~blistog/xhtml.html HTTP/1.1
14	0.014865	79.171.82.147	158.36.131.51	TCP	80 > 12157 [ACK] Seq=1 Ack=710 Win=7090 Len=0
15	0.032256	79.171.82.147	158.36.131.51	TCP	[TCP segment of a reassembled PDU]
16	0.032283	79.171.82.147	158.36.131.51	HTTP	HTTP/1.1 200 OK (text/html)
17	0.032308	158.36.131.51	79.171.82.147	TCP	12157 > 80 [ACK] Seq=710 Ack=1729 Win=65535 Len=0
18	0.032325	79.171.82.147	158.36.131.51	TCP	80 > 12157 [FIN, ACK] Seq=1729 Ack=710 Win=7090 Len=0
19	0.032342	158.36.131.51	79.171.82.147	TCP	12157 > 80 [ACK] Seq=710 Ack=1730 Win=65535 Len=0
21	0.059238	158.36.131.51	79.171.82.147	TCP	12157 > 80 [FIN, ACK] Seq=710 Ack=1730 Win=65535 Len=0
22	0.060711	79.171.82.147	158.36.131.51	TCP	80 > 12157 [ACK] Seq=1730 Ack=711 Win=7090 Len=0

Figur 10: TCP- og HTTP-transaksjon i Wireshark

De tre første pakkene er de som foretar handshaken. De tre neste er http-meldinger som ber om html-siden og får dem overført. Så følger en enkelt kvitteringspakke fra klienten. Og til slutt følger de fire TCP-segmentene som benyttes for å kople ned forbindelsen.



Figur 11: : "Flow Graph" for TCP-forbindelsen i Fig 4.

Dersom vi tar en nærmere titt på innholdet i TCP-headeren så vil vi se at denne består i utgangspunktet av 20 byte, men med mulighet for å legge til ekstrainformasjon i Options-feltet.

TCP Header

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port																Destination port															
32	Sequence number																															
64	Acknowledgment number																															
96	Data offset				Reserved				CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
128	Checksum																Urgent pointer															
160 …	Options (if Data Offset > 5) …																															

Figur 12: TCP-headeren (kilde: Wikipedia)

Kilde- og mål-portnummer brukes på nøyaktig samme måte som i UDP. Det samme gjelder sjekksummen. I mange mange nyere implementeringer av TCP brukes for øvrig ikke sjekkesummen fordi tilsvarende sjekksummer på nettverks- og link-nivå gjør samme nytte.

Headeren inneholder flere enkeltbit som benyttes som flagg. Dvs de ”flagger” hvordan andre felt i headeren skal behandles, eller angir spesialinformasjon om hvilken rolle et segment har i forløpet fra oppstart til slutt. De viktigste flaggene i forbindelse med vanlig bruk er ACK, SYN, FIN og RST. SYN, FIN og RST har alle med oppkopling og nekopling av logiske forbindelser å gjøre, mens ACK-flagget sier hvorvidt kvitterings-nummeret er gyldig (bør tas hensyn til) eller ikke.

Ved oppstart benyttes flaggene SYN og ACK. I det første segmentet fra klienten er SYN-feltet satt til 1, ACK til 0. SYN kan leses som SYNkronisering. Poenget med ”handshaken” er nettopp å oppnå en felles forestilling om hva som foregår og for å bli kjent med hverandres evner. I svaret fra tjeneren kan enten bare RST-flagget være satt. I så fall er det ingen prosesser som kjører bundet til portnummert, og forbindelsen blir avslått og avsluttet der. Det vanligste er dog at tjeneren svare med en pakke der både SYN- og ACK-flaggene er satt til 1. Så avsluttes handshaken med en kvitteringsmelding med bare ACK-flagget satt fra klienten.

Sekvensnummeret er ett 32-bits binærtall. Dette telles opp med antall byte med data som har blitt sendt. Både klient og tjener starter på et rimelig tilfeldig valgt tall hver. Kvitteringsnummeret er alltid sist mottatte sekvensnummer fra kommunikasjonspartneren pluss 1. Dvs at dersom klienten sender et segment med sekvensnummer 157 som ikke inneholder noen data til tjeneren, så vil denne svare med kvitteringsnummer 158. Dersom klienten sender et segment med sekvensnummer 157 som innholder 200 byte med data, så vil tjeneren benytte kvitteringsnummer 157+200+1=358.

For å sørge for at klient og tjener ikke sender hverandre flere pakker enn de har satt av plass til i datamaskinens minne, så benyttes en kontinuerlig signalering av ”Window Size”. I fig 4 ser vi at klienten starter med å si at den har satt av en

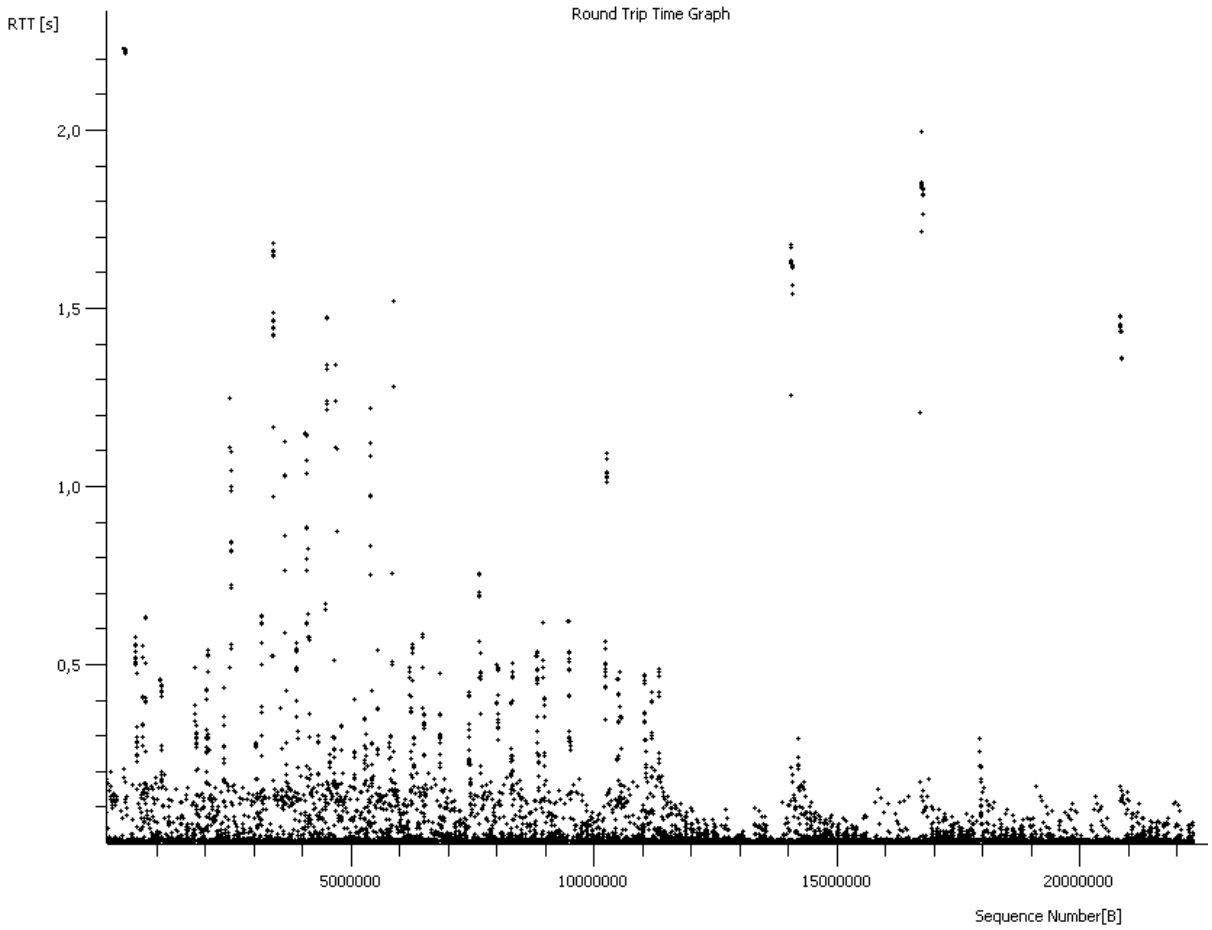
kapasitet på 65535, mens tjeneren svarer med at den har satt av 5840. I løpet av en større dataoverføring vil begge parter justere opp og ned disse. Dersom en av partene signalerer 0, betyr det at partneren må vente til det er plass hos den andre igjen. Dette kalles *flytkontroll* (”flow control”) og målet er å samordne senderens skrivehastighet med leserens lesehastighet.

Options-feltet gir mulighet for å utveksle mer spesialisert informasjon. I fig 4 ser vi at klienten signalerer MSS = 1460, mens tjeneren svarer at den har MSS = 1380. MSS står for Maximum Segment Size og angir hvor mye data det er mulig å legge inn i et segment. Poenget med å utveksle denne informasjonen er at dersom klienten nå begrenser seg til å aldri legge mer enn 1380 byte meldinger inn i TCP-segmentet, så vil disse overføres til tjeneren uten å måtte splittes opp i flere mindre pakker på veien. Dette sparer kapasitet på tjeneren som slipper å sette fragmentene sammen igjen og sikrer dermed bedre responstid.

Nedkoplingen av TCP-forbindelsen foregår ved at en av partene sender et segment der FIN-flagget er satt og får kvittering på den. Forbindelsen er etter det ”halv-åpen”. Den lukkes først når den andre parten også sender et segment der FIN-flagget er satt, og mottar en kvittering på den.

Spørsmålet om når segmenter skal sendes på nytt fordi det ikke har kommet kvittering etter rimelig tid behandler klient og tjener uavhengig av hverandre. For å finne hva timeout-perioden skal være registreres kontinuerlig hvor lang tid det tar fra en pakke er sendt til kvittering er mottatt: RTT (Round Trip Time). Timout-perioden settes med utgangspunkt i disse målingene pluss målt variasjon i målingene (”jitter”). Timeout-peroden tilpasses dermed hele tiden tilstanden i nettverket. Segmenter sendes på nytt når kvittering ikke er mottatt innen timout, eller det mottas tre kvitteringer på ett segment med lavere sekvensnummer.

Figuren under viser variasjonen i RTT i en overføring av en 23 MB fil fra en server hos Cisco i USA til en PC i Oslo:



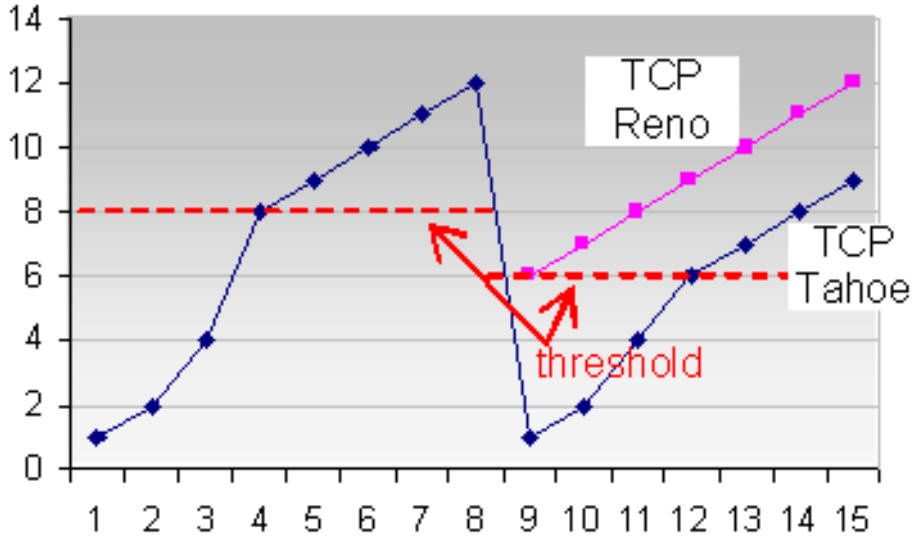
Figur 13: Variasjon i RTT i følge Wireshark

Responstiden kan i følge disse målingene variere mellom ca 2,2 sekunder og et fåtall millisekund for denne overføringen. Når vi da tenker tilbake på hva vi så i forrige modul om årsaker til forsinkelser så virker de laveste RTTene umulige, men de må ses i sammenheng med bruk av Web-cache/Proxy-server.

Metningskontroll

TCP har også en mekanisme som skal ta hensyn til tilstanden i nettverket mellom kommunikasjonspartene. Flytkontroll tar hensyn til kapasiteten på endesystemene, metningskontroll (”congestion control”) er et forsøk på å ta hensyn til kapasitet i, og belastning på, hele nettverket mellom partene.

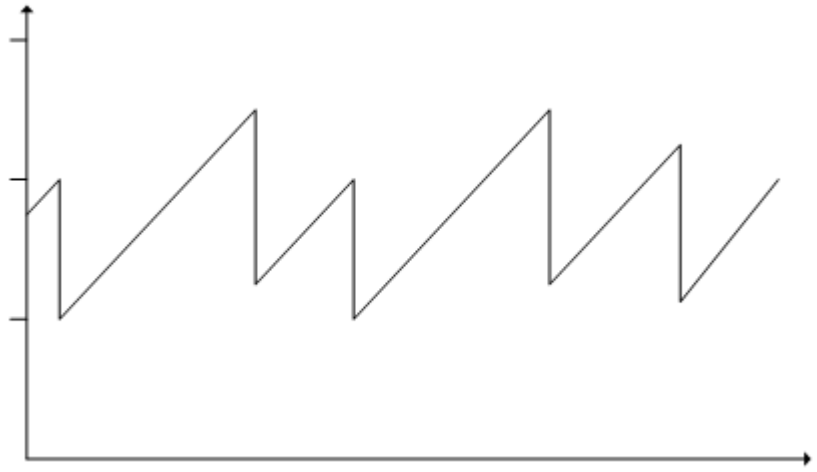
Prinsippene bak metningskontrollen er å starte med en utforsking av grensen for hvor pakketap i nettverket ligger og deretter forsøke å holde seg tett under denne. For alle praktiske formål betyr det at TCP begynner med å sende et segment og venter til det har kommet kvittering på dette. Så dobles det til to, dobles igjen (til 4), dobles igjen (til åtte), osv. til pakketap forekommer, eller man når partners signaliserte vindusstørrelse. Ved pakketap halveres metningsvinduet. Etter dette undersøker man på nytt max overføringskapasitet, men denne gangen dobler man ikke størrelsen på vinduet. Det øker typisk med ca 1 MSS pr runde inntil grensen er nådd igjen, for så å ta en ny reduksjon. For to ulike versjoner av TCP er dette illustrert i figuren under:



Figur 14: TCP Tahoe (eldst) vs TCP Reno (nyere)

Det finnes ulike impelementeringer av TCP i ulike operativsystsmer, og disse utfører metningskontrollen på forskjellige måter. Alle har likevel som mål at de aldri skal ”presse ut” andre brukere, men oppnå en bitrate som er så god som mulig uten at det blir ”urettferdig” bra i forhold til andre brukere.

Den typiske profilen på en TCP-overføring er da noe a la den du ser i figuren under:



Figur 15: Størrelse på metnings-vindu vs tid

Kontrollspørsmål til kapittel 3

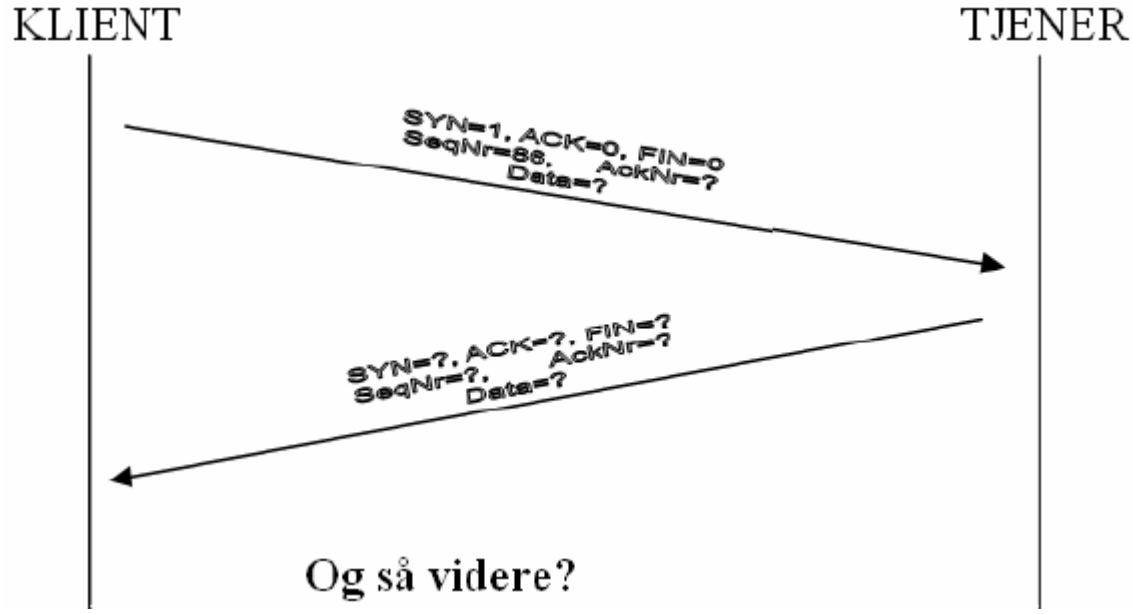
8. Hva brukes portnummer til?
9. Hva er den viktigste forskjellen på UDP og TCP?

10. En TCP-header har et felt for header-lengde. Hvorfor har ikke en UDP-header dette?

11. Definer multipleksing/demultipleksing. Hvordan foregår mux/demux i TCP/IP-stacken?

12. Hva er likhetene og forskjellene på TCP- og UDP-protokollene?
    Hvorfor er begge i bruk?
    Hvilken vil det være naturlig å bruke ved IPtelefoni (begrunn svaret).

13. En telnet-klient skal kople seg opp til en telnet-server for å oversende
    bokstaven ”A” og deretter avslutte forbindelsen. Du kan gå ut fra at klienten starter med sekvensnummer (SeqNr) 86 og tjeneren med sekvensnummer 23.
    Vis ved å fullføre figuren under hvordan TCP-forbindelsen settes opp,
    dataoverføringen foretas, og nedkoplingen foregår. (Figuren under viser bare starten på forbindelsen. Erstatt spørsmålstegnene i figuren med dine forslag til korrekte verdier.)



14. Forklar hvordan flytkontroll foregår i TCP.

15. Forklar hvordan metningskontroll foregår i TCP.

16. Hva betyr flytkontroll og metningskontroll for hvilken bitrate du kan oppnå i TCP-overføring av data?

Studieenhet 3: *Nettverks- og Link-laget*

Mål for studieenhet 3

Læreplanens mål

Mål for opplæringen er at eleven skal kunne

- beskrive datagram routing fungerer
- finne routingtabell for en router i et lite nettverk etter Dijkstras algoritme
- definere Autonomt System (AS), samt forklare forskjellen mellom inter-AS og intra-AS routing
- forklare hva et IP-nettverk er med utgangspunkt i begrepene IP-adresse, nettmaske og default gateway; her under subnetting, CIDR og klasse A-, B- og C-nettverk
- beskrive funksjonsmåten til IP-, NAT-, DHCP- og ICMP-protokollene.
- forklare forskjellene mellom IP v4 og IP v6
- definere og forklare hvilke tjenestetyper som er tilgjenglige i nettverkslaget, samt knytte disse til Ethernet II (IEEE 802.3) headeren
- kjenne til mulighetne for feiloppdagelse og -håndtering med utgangspunkt i teknikkene: paritetsbit, todimensjonale paritetsbit og CRC
- forklare bruk av ARP-protokollen i relasjon til MAC-adresser og IP-nettverk
- forklare oppbyggingen av en Ethernet-ramme, hvordan denne kodes ut på linjen, og hvordan kollisjonshåndteringen fungerer
- beskrive funksjonsmåten til komponentene hub, bro, svitsj og router i forhold til Ethernet og overliggende lag i TCP/IP-modellen
- kjenne de viktigste forskjellene mellom trådbundet Ethernet (IEEE 802.3) og trådløst (IEEE 802.11 protokollene)
- bruke IPCONFIG, PING, NETSTAT, ROUTE og ARP kommandoene til å undersøke nettverksparametre og feilsøke nettverk.

Kapittel *4 - Nettverkslaget*

Nettverkslaget har i henhold til TCP/IP-modellen ansvar for routingen (veivalgene) av datagrammer fra avsender til mottager i Internett.
Vi skal her hovedsakelig konsentrere oss om de aspektene ved nettverkslaget som er tilgjenglige lokalt. Protokollene og programvaren som støtter applikasjonslags- og transportlags-tjenstene befinner seg i all hovedsak på endesystemene i nettverket. Når vi kommer ned på nettverks- og link-laget er de viktigste protokollene og standardene til stede på de fleste komponentene som en pakke besøker på veien fra avsender til mottager.

Routing: verktøyene tracert og ipconfig

For å se hvilken vei en datapakke (datagrammet) følger gjennom Internett kan vi bruke verktøyet tracert. Figuren under viser veien pakken følger på routerene fra en maskin i Oslo til en server i Finland:

```
C:\>tracert www.eunetip.net

Sporer rute til www.eunetip.net [62.142.74.10]
over maksimalt 30 hopp:

  1     2 ms    <1 ms     *     stolav-gw4.uninett.no [158.36.84.169]
  2   <1 ms    <1 ms    <1 ms  stolav-gw1.uninett.no [158.36.84.161]
  3     1 ms    <1 ms    <1 ms  oslo-gw.uninett.no [128.39.65.21]
  4    10 ms     15 ms     10 ms  nix.eunetip.net [193.156.90.30]
  5    16 ms     16 ms     16 ms  as0-0.bbr2.hel1.fi.eunetip.net [213.192.191.209]
  6    17 ms     16 ms     16 ms  ae0-10.heltli-gw1.fi.elisa.net [213.192.191.50]
  7    17 ms     17 ms     16 ms  tg9-1.rx9.alto.esp.fi.eunetip.net [139.97.6.154]
  8    16 ms     16 ms     16 ms  saunalahti.fi [62.142.74.10]

Soking fullfort.
```

Figur 16: tracert-utskrift Oslo -> Finland

Hvilken vei pakken følger er bestemt av IP-adressen. I dette tilfellet ser vi at måladressen til serveren med DNS-navn www.eunetip.net var 62.142.74.10 (jf Figur 17)

```
C:\>nslookup www.eunetip.net
Server:  ns2.nith.no
Address:  158.36.131.4

Ikke-autoritativt svar:
Navn:    www.eunetip.net
Address:  62.142.74.10
```

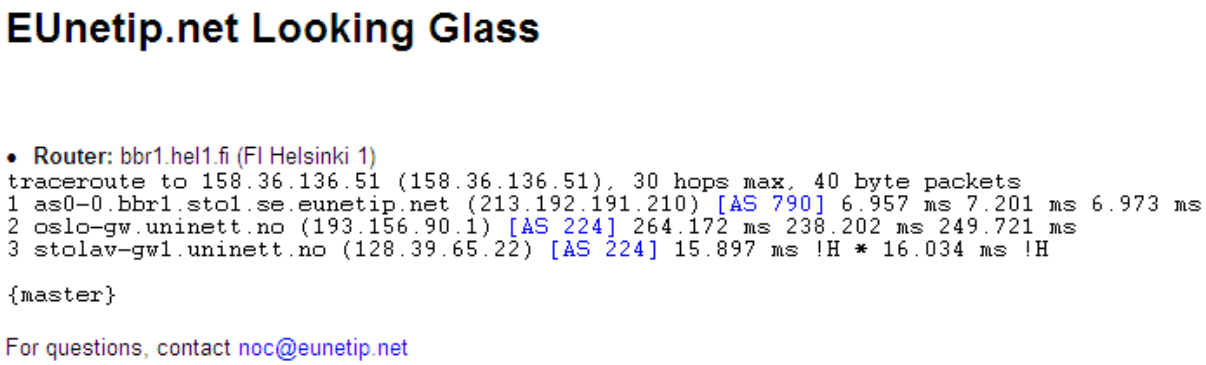
Figur 17: nslookup av www.eunetip.net

Maskinen som sendte tracert-forespørselene var den jeg satt på (158.36.131.51). For denne finner vi de viktigste nettverksparametrene ved hjelp av kommandoen ipconfig (se Figur 18):

```
C:\>ipconfig

Windows IP-konfigurasjon

Ethernet-kort eth0:

        Tilkoblingsspesifikt DNS-suffiks : nith.no
        IP-adresse . . . . . . . . . . . : 158.36.131.51
        Nettverksmaske . . . . . . . . . : 255.255.255.128
        Standard gateway . . . . . . . . : 158.36.131.1
```

Figur 18: ipconfig-kommandoen skriver ut den grunnleggende lokale IP-konfigurering

Internett er et nettverk av nettverk med ulike eiere og dermed også forskjellige administratorer. Sammenkoplingen av disse nettverkene skjer gjennom det som kalles Peering. I eksempelet over så er det to forskjellige nettverkseiere med i bildet. UNINETT (som drifter det norske nettverket for universiteter og høyskoler), og EUnet (som i denne sammenhengen er en finsk ISP). Dersom vi går inn på EUnet Finland sin "Looking Glass Server" (http://www.eunetip.net/look.shtml) kan vi utføre en tracert i motsatt retning:

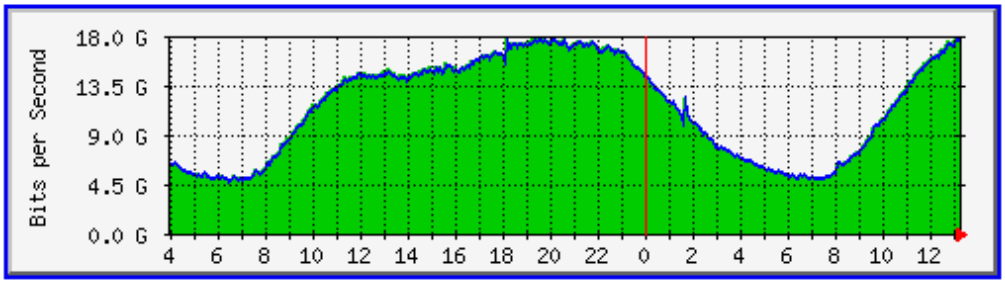


Figur 19: tracert Finalnd -> Norge

Det er verdt å merke seg at denne ikke viser like mange trinn som den vi foretok fra egen maskin. I tillegg viser den ikke bare IP-adressene til routere, men også AS-nummer.

Sett fra øverste nivå av består Internett av Autonome Systemer. Et Autonomt system (AS) er ett eller flere IP-nettverk som er under en felles administrativ ledelse. Mellom AS (inter-AS routing) benyttes alltid routingprotokollen BGP, innenfor ASet (intra-AS routing) står fritt for eier/administrator å velge routing-protokoll. Dette kalles også for hierarkisk routing fordi vi benytter andre routing-prinsipper på øverste nivå, enn vi gjør på lavere. Mellom ISPer vil f.eks pris og hvem man har peering-avtaler med spille en like viktig rolle som båndbredde, avstand og andre ytelsesbaserte kriterier.

I Norge er den eldste peering-sentralen NIX (Norwegian Internet eXchange) administrert av UNINETT/Universitetet i Oslo (se http://www.uio.no/nix/). Svært mye av Internett-trafikken går faktisk gjennom noen får switcher.



Figur 20: Statistikk for gjennomstrømning ved NIX

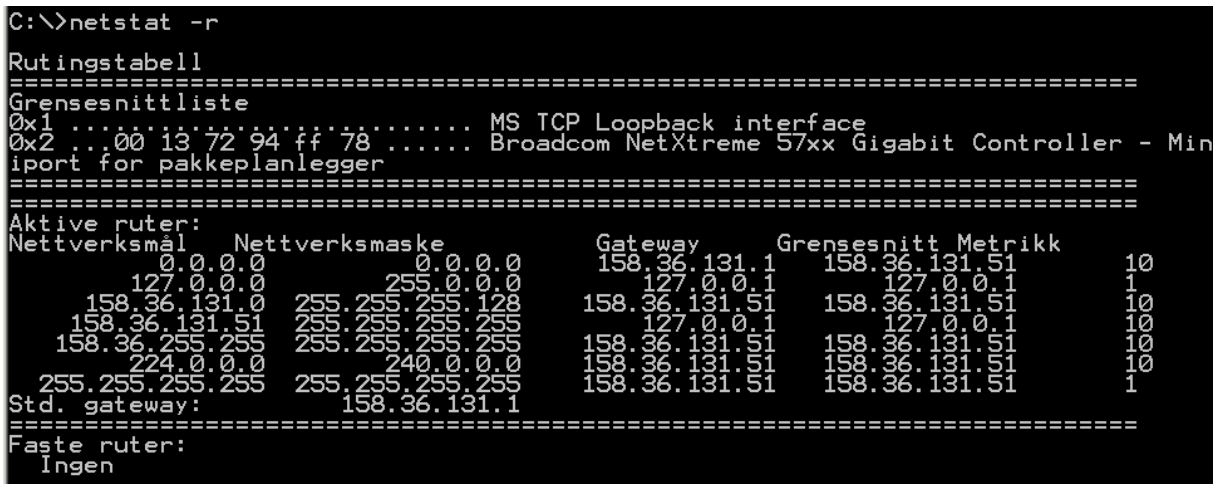
Tildeling av IP-adresser og AS-nr er det organisasjonen RIPE (http://ripe.net) som står for i Europa. De administrerer også de europeiske DNS-navnerommene. Desom du lurer på hvem som eier en bestemt IP-adresse så kan du finne ut av dette ved et WHOIS-oppslag mot RIPE sin database (se utdrag under). Det er også RIPE ISPer og andre må henvende seg til for å få tildelt IP-adresser.

```
inetnum:        158.36.0.0 - 158.36.255.255
remarks:
remarks:        This inetnum has been transfered as part of the ERX.
remarks:        It was present in both the ARIN and RIPE databases, so
remarks:        the information from both databases has been merged.
remarks:        If you are the mntner of this object, please update it
remarks:        to reflect the correct information.
remarks:
remarks:        Please see the information for this process:
remarks:        http://www.ripe.net/db/erx/erx-ip/network-158.html
remarks:
remarks:        **** INFORMATION FROM ARIN OBJECT ****
remarks:        netname: UNINETT1
descr:          Uninett
descr:          UNINETT AS
descr:          Trondheim
descr:          7465
remarks:        country: NO
admin-c:        PK1847-RIPE
admin-c:        HE386-RIPE
tech-c:         UNUN4-RIPE
remarks:        remarks:     Uninett South & East Norway
remarks:        abuse e-mail: <abuse@uninett.no>, phone: +47 73 55 79 60
```

Figur 21: Utdrag fra RIPE databasen (oppslag 158.36.131.1)

Internett-protokollen: IPv4 og routing

Routing starter på din egen maskin, som har sin egen routingtabell. Routingtabellen omtales også som forwarding-tabellen fordi det er denne som angir hvilket grensesnitt/adapter/nettverkskort datagrammet med en bestemt mottager IP-adresse som skal sendes ut på (”forwardes på”). For å få opp den lokale routing-tabellen benytter man kommandoen `route print`, eller `netstat -r`.



Figur 22: Lokal routing-tabell

For å kunne lese routing-tabellen er det flere ting vi trenger å vite:

1. IP-protokollen benytter **prefix-routing**. IP-adressen består av to hoved-deler som skilles ad ved hjelp av nettmasken.
2. Standard Gateway er IP-adressen alle datagram med annen mål-adresse enn de som befinner seg i routing-tabellen skal sendes til. Det er dermed veien ut av det lokale IP-nettverket.
3. IP-adressen er egentlig ikke adressen til en maskin, men adressen til et grensesnitt/adapter/nettverkskort på maskinen.
4. Det finnes en del IP-adresser som har spesiell betydning.

En IP (v4) adresse er 32 bits tall. Det betyr at IP-adressen 158.36.131.51 er bare en annen måte å notere det desimale sifferet 2653193011 på. IP-adressen som routingen foregår ut fra er 10011110 10011110 10000011 00110011. Den vanlige måten å oppgi IPv4-adresser på er dermed å skrive opp hver av de fire bytene i adressen som et desimaltall mellom 0 og 255 adskilt med punktum.

IP-adressen består av to hoveddeler: nettverksprefix og host-del. Hva som er hva er det nettverksmaskin som bestemmer. Nettverksprefix skilles ut ved at det er de binære sifrene der nettverksmasken består bare av 1'ere. Host-delen er der nettverksmasken er bare 0'er. Når nettverksmasken er 255.255.255.128, tilsvarer det 11111111 11111111 11111111 10000000. Dermed er nettverksprefixet de 25 første binærsifrene i IP-adressen, mao 10011110 10011110 10000011 0, mens host-delen er 0110011. Nettverksprefixet er felles for alle adapter/maskiner/nettverkskort som tilhører samme IP-nettverk. Prefixet fungerer da litt på samme måte som etternavnet gjør innefor en familie, mens host-delen tilsvarer fornavnet.

	Nettverksprefix																								Host							
Adresse	1	0	0	1	1	1	1	0	1	0	0	1	1	1	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1
Maske	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

Figur 23: Nettverksprefix og Host for 158.36.131.51/25

IP-adressen 158.36.131.51 med nettmaske 255.255.255.128, kan også skrives 158.36.131.51 /25. Dette kalles CIDR-notasjon. /25 angir da nøyaktig det samme som nettmasken, nemlig at prefixet er de 25 innledende binærsifrene.

I IP-nettverket der 158.36.131.51 /25 befinner seg er det da mulig å bruke adressene mellom 158.36.131.1 til 158.36.131.126 på ulike adapter/maskiner/nettverkskort. 158.36.131.0 er ikke lov å bruke. 158.36.131.127 er den høyeste mulige adressen og brukes i hehold til IPv4-protokollen til **broadcast** ("kringkasting") innefor dette IP-nettverket. Det betyr at IP-datagram med denne mottager-adressen sendes til alle adapter/maskiner/nettverkskort innenfor nettverket.

På www kan du finne en mengde forskjellige subnet-kalkulatorer som vil hjelpe deg i å konvertere mellom desimal og binær IP-adresse, finne prefix og lignende. For eksempel http://www.subnetmask.info/

Routing i Internett forgår da ut fra to ulike tall. Nettverksprefix og AS-nummer.

Noen andre IP-adresser som skal behandles på en spesiell måte er:

- Adresser der første byte er 127. For eksempel 127.0.0.1 Dette er adressen til "localhost". Altså maskinen som du sitter på. Omtales også som loopback.
- Adressene 10.0.0.0 /8, 172.16.0.0 /12 og 192.168.0.0 /16 er holdt av til "private nettverk" og er ikke route-bare i Internett. Disse brukes oftest bak en NAT-tjener (se under).
- 169.254.0.0 /16 benyttes også kun lokalt, vanligvis ved "selvkonfigurering". De fleste vil kun se disse adressene dersom maskinen ikke får kontakt med en DHCP-tjener (se under).
- Adresser der første byte ligger mellom 224 og 254 skal også behandles på en spesiell måte, og vil ikke tildeles enkelt-maskiner. Noen av dem brukes til multicast, andre er foreløpig ikke tatt i bruk i det hele tatt.

Opprinnelig var IP-adressene delt opp i tre ulike klasser: A, B og C. A var adressene der første byte var 0-127, B 128-191, C 192-223. Mens klassefull adressering ble brukt opererte man kun med nettverksprefixer på 1 (A), 2 (B), eller 3 (C) hele byte. Dette systemet har nå mest historisk interesse. Det er likevel verdt å merke seg at i routing-tabellen over (Figur 22) så er en mulig broadcast-adresse 158.36.131.51 /25, oppgitt som 158.36.255.255 /24 fordi dette ville vært broadcast-adressen for klasse B nettverket.

Dersom vi nå ser på routing-tabellen i Figur 22, så skal den leses fra bunnen og oppover. Den starter med den meste spesifikke adressen, den der flest bit er "låst" ved hjelp av nettmasken.

Standard Gateway er IP-adressen til den routeren som leder ut av det IP-nettverket du befinner deg innefor. Dersom ikke denne er satt opp vil du ikke ha tilgang til Internett, kun ditt lokale IP-nettverk.

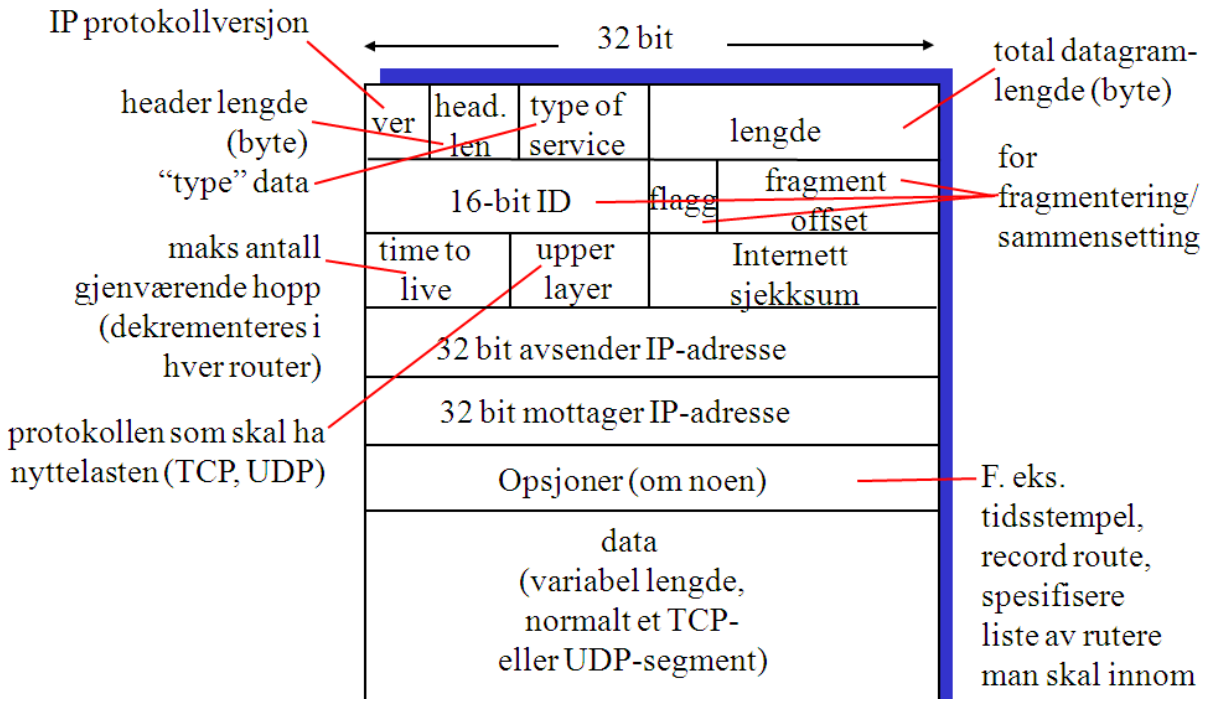
IPv4 headeren

Som vi har sett i forbindelse med andre protokoller kan vi finne ut mer om hvordan IP protokollen er bygd opp og virker ved å se nærmere på headeren.

```
Internet Protocol, Src: 158.36.131.51 (158.36.131.51), Dst: 195.139.129.130 (195.139.129.130)
  Version: 4
  Header length: 20 bytes
⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 92
  Identification: 0xcab3 (51891)
⊞ Flags: 0x00
  Fragment offset: 0
⊞ Time to live: 1
  Protocol: ICMP (0x01)
⊞ Header checksum: 0x8888 [correct]
  Source: 158.36.131.51 (158.36.131.51)
  Destination: 195.139.129.130 (195.139.129.130)
```

Figur 24: En IPv4 header fra Wireshark (hentet fra en tracert-sesjon)

Som vi ser av figuren under består IP-headeren i likhet med TCP-headeren av 20 byte med mulighet til å legge inn ekstra "Options".



Figur 25: IPv4 headerens oppbygging

Rollene til de ulike feltene i headeren er som følger:

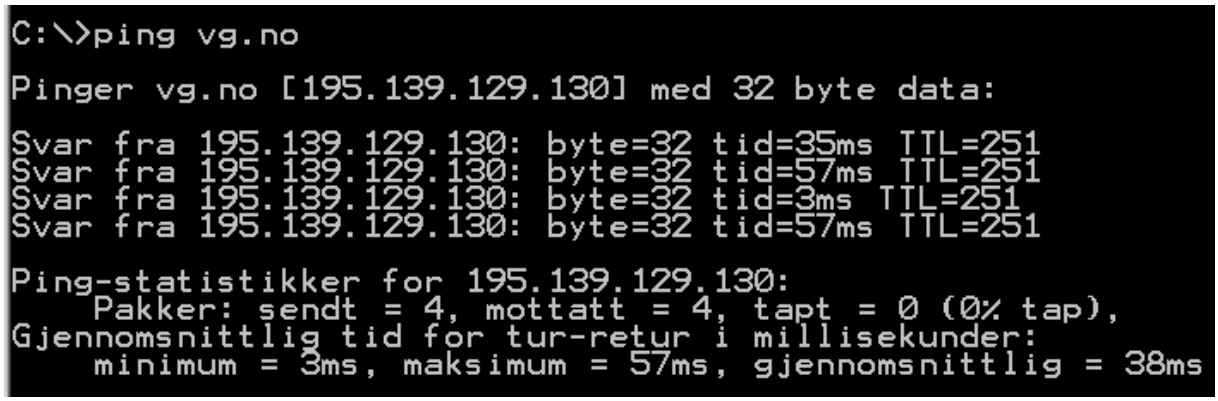
- Ver-feltet er på 4 bit. I IPv4 er dette da alltid satt til 0100.

- Head-length feltet er på 4 bit og oppgir hvor lang headeren er i blokker på 4 byte. Dersom det ikke er satt noen Options vil denne ha desimal verdi 5 (binært: 0101)
- Type of Service (ToS) feltet kalles ogs DiffServ-feltet, er på en byte, og skal gi muligheten til å styre nærmere hvordan routerene behandler og prioriterer IP-datagram. Brukes lite, om i det hele tatt. Internett er ”best effort” fremdeles.
- Lengde-feltet er på 16 bit og angir hvor mange byte datagrammet er på. Maxverdien er 65535 byte.
- De tre neste feltene (ID, flagg og fragment offset) gir routere muligheten til å dele ett IP-datagram som er for stort i forhold til link-protokollen som forbinder routeren med neste router. Disse feltene kan da brukes dersom datagrammet blir delt opp flere mindre datagrammer. Stort sett sette Don’t Fragment flagget slik at man får en tilbakemelding fra routeren som ikke kan håndtere størrelsen man bruker og så nedskalerer man i forhold til tilbakemeldingen.
- TTL (Time To Live) feltet er på en byte og angir hvor mange routere datagrammet kan besøke før det skal droppes. TTL telles ned med en for hver router IP-pakken har vært innom. Motivasjonen er å unngå at pakker som routes feil (”sirkulært”) ikke skal kunne leve evig. Routeren som da teller ned feltet til 0 skal droppe datagrammet og sende en feilmeding (ICMP-pakke) tilbake til avsender. Når vi bruker tracert så lager vi først et IP-datagram der dette feltet er satt til 1, så 2, osv.
- Protocol-feltet er på en byte og brukes til å angi en tallkode for hvilken protokoll innholdet i datagrammet hører til. ICMP har kode 1, TCP 6, UDP 17 osv. Dette feltet brukes altså til mux/demux (se forrige studie-enhet)
- Header sjekksummen er laget etter samme regler som for TCP og UDP: Hver linje i headeren summeres binært, og deretter inverteres bit’ene. Under utregningen settes feltet til 0. Pga nedtellingen av TTL-feltet må sjekksumme beregnes på nytt i hver router lansmed ruten. Dersom den er feil, droppes datagrammet.
- Avsenders og mottagers IP-adresse er på 32 byte hver og identifiserer disse unikt i Internett. Routeren benytter mottager-adressen til å slå opp i sin routingtabell og så velg hvilket adapter/nettverkskort/port datagrammet skal sendes videre på.

ICMP

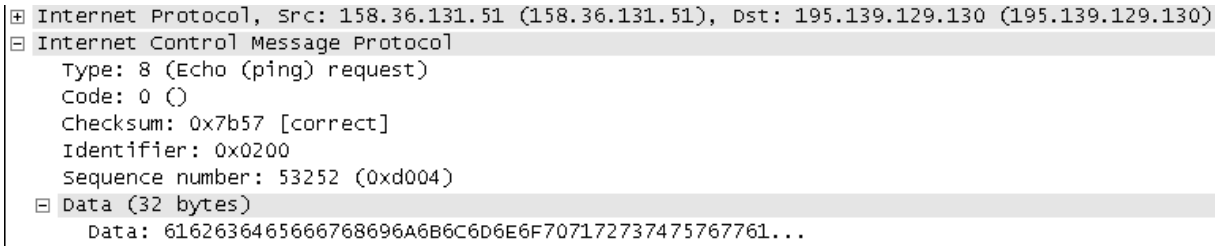
Internet Control Message Protocol (ICMP) er opprinnelig laget for å gi tilbakemelding til avsender og avsenders router om feil i IP-datagram. Den kan også brukes til å signalisere at det finnes en billigere vei innenfor et IP-nettverk, men det er ikke lenger en vanlig anvendelse.

Når vi bruker ping for å se om det er mulig å nå en IP-adresse så er det ICMP-pakker vi sender til mottageren.



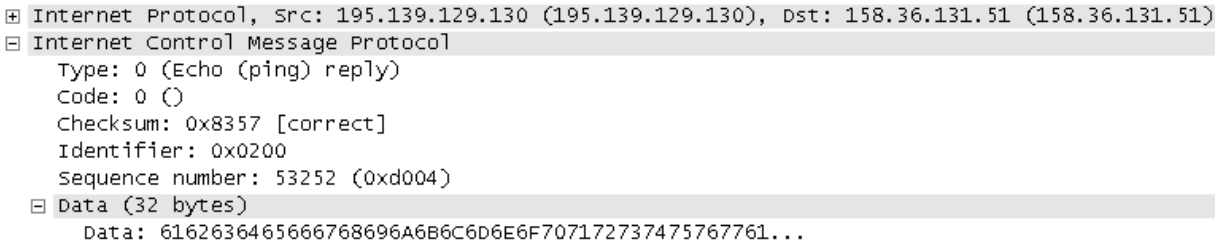
Figur 26: ping vg.no

Windows sender standard 4 ICMP echo-pakker. Echo-pakker er der type-feltet i ICMP-headeren er satt til verdien 8.



Figur 27: ICMP ping forespørsel (Wireshark)

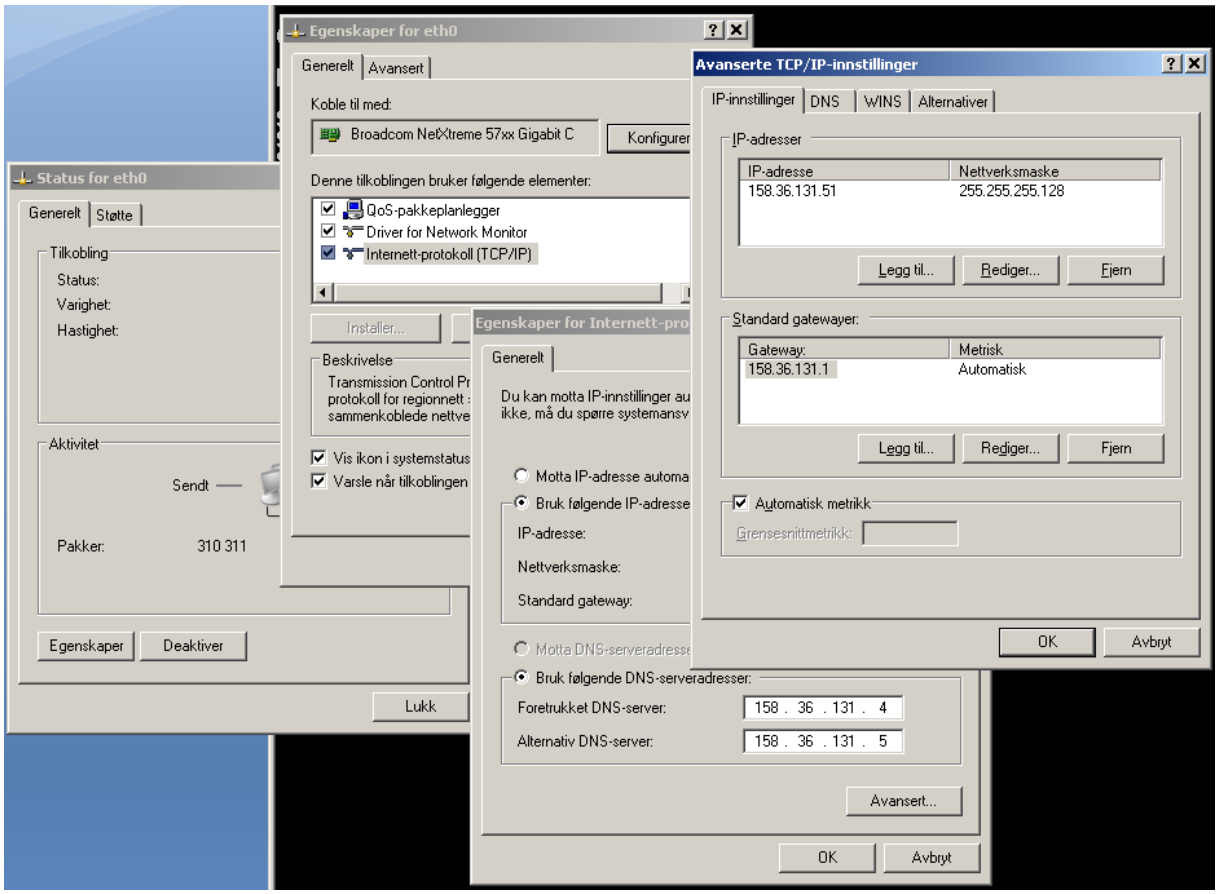
I svar-pakken vi når serveren er oppe og går, så er type-feltet satt til 0.



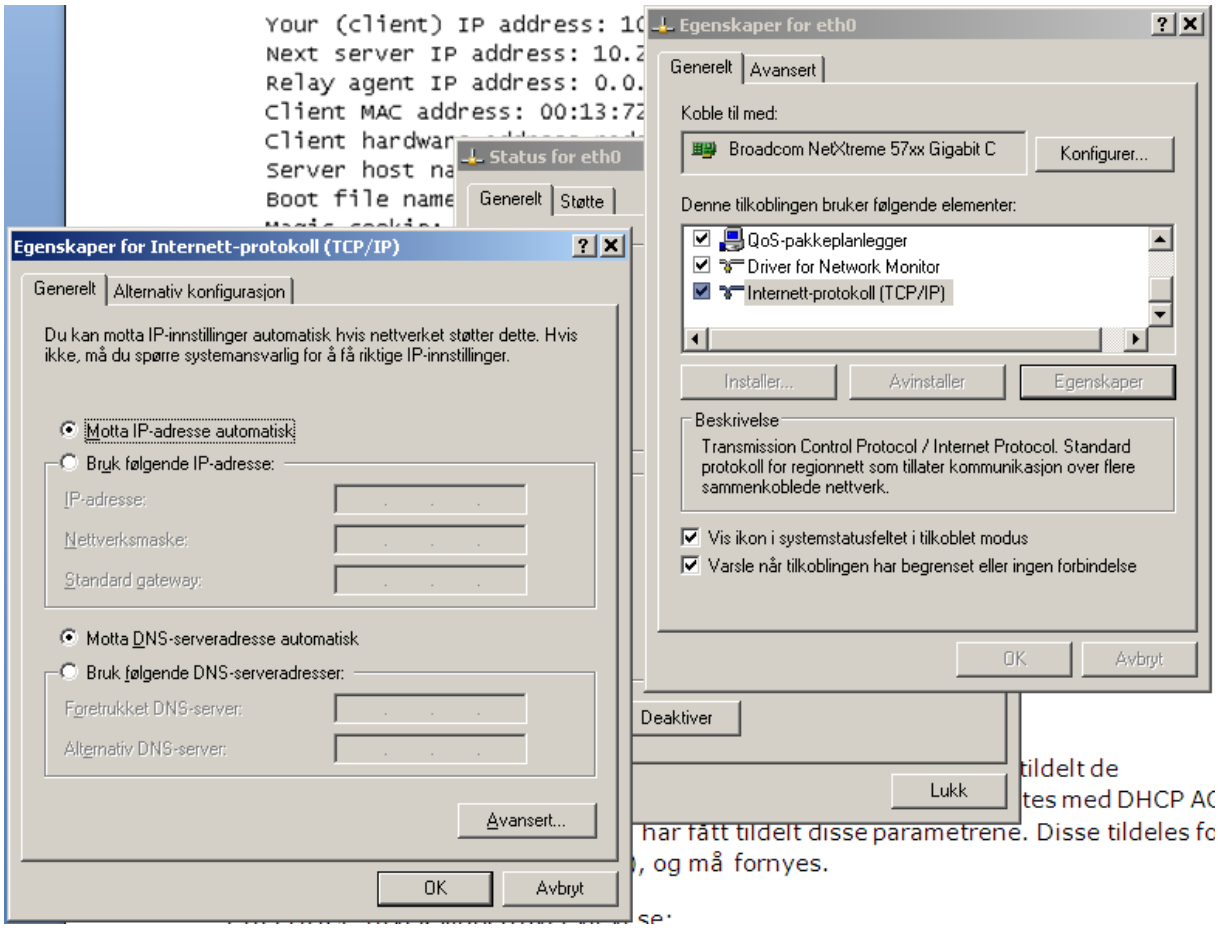
Figur 28: ICMP ping svar

Automatisk IP-oppsett: DHCP

IP-adresse, nettmaske og standard gateway kan settes manuelt (”statisk”), men i de fleste hjemme- og bedrifts-nettverk er det typisk at man bruker en DHCP-tjener til å dele ut nettverksparametere (IP-adresse, nettverksmaske, standard gateway, DNS-tjenere, med mer) dynamisk.



Figur 29: Statisk oppsett av IP under Windows XP



Figur 30: Windows XP oppsett for å bruke DHCP

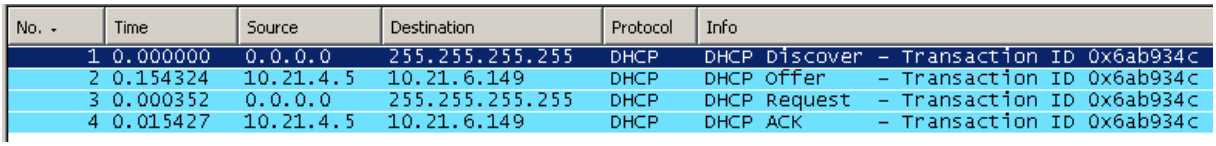
Måten DHCP benyttes på er at det finnes en DHCP-tjener på nettverket. Denne kontaktes av maskinen din etter lastingen av operativsystemet. Før det har den ikke satt noen verdier for IP-nettverket:

```
C:\>ipconfig

Windows IP-konfigurasjon

Ethernet-kort eth0:

        Tilkoblingsspesifikt DNS-suffiks : nith.no
        IP-adresse . . . . . . . . . . . : 0.0.0.0
        Nettverksmaske . . . . . . . . . : 0.0.0.0
        Standard gateway . . . . . . . . :
```

Figur 31: Uten IP-konfigurasjon

Når maskinen starter opp nettverkskonfigureringen sendes først en DHCP-discover pakke. Siden man ikke selv har noen IP-adresse og ikke vet DHCP-tjeneren sin så sendes dette som broadcast: avsender-IP= 0.0.0.0, mottager-IP=255.255.255.255. (I Wireshark må man benytte filteret bootp for å få frem

kun DHCP-pakker. Dette har den historiske årsak at DHCP- er en videreutvikling av BOOTP-protokollen).



No.	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0x6ab934c
2	0.154324	10.21.4.5	10.21.6.149	DHCP	DHCP Offer - Transaction ID 0x6ab934c
3	0.000352	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0x6ab934c
4	0.015427	10.21.4.5	10.21.6.149	DHCP	DHCP ACK - Transaction ID 0x6ab934c

Figur 32: DHCP tildeling av IP-parametre (Wireshark)

DHCP-tjeneren svarer med å tilby et oppsett i form av en DHCP Offer pakke der verdiene er spesifisert.

```
Bootstrap Protocol
  Message type: Boot Reply (2)
  Hardware type: Ethernet
  Hardware address length: 6
  Hops: 0
  Transaction ID: 0x06ab934c
  Seconds elapsed: 0
⊞ Bootp flags: 0x0000 (Unicast)
  Client IP address: 0.0.0.0 (0.0.0.0)
  Your (client) IP address: 10.21.6.149 (10.21.6.149)
  Next server IP address: 10.21.4.131 (10.21.4.131)
  Relay agent IP address: 0.0.0.0 (0.0.0.0)
  Client MAC address: 00:13:72:94:ff:78 (00:13:72:94:ff:78)
  Client hardware address padding: 00000000000000000000
  Server host name not given
  Boot file name: pxelinux.0
  Magic cookie: (OK)
⊞ Option: (t=53,l=1) DHCP Message Type = DHCP Offer
⊞ Option: (t=54,l=4) DHCP Server Identifier = 10.21.4.5
⊞ Option: (t=51,l=4) IP Address Lease Time = 2 hours
⊞ Option: (t=1,l=4) Subnet Mask = 255.255.252.0
⊞ Option: (t=15,l=7) Domain Name = "nith.no"
⊞ Option: (t=3,l=4) Router = 10.21.4.1
⊟ Option: (t=6,l=8) Domain Name Server
    Option: (6) Domain Name Server
    Length: 8
    Value: 9E2483049E248303
    IP Address: 158.36.131.4
    IP Address: 158.36.131.3
⊞ Option: (t=44,l=4) NetBIOS over TCP/IP Name Server = 10.21.4.131
  End Option
  Padding
```

Figur 33: DHCP Offer

Din egne maskin svarer med en DHCP Request, der den ber om å få tildelt de nettverksparametrene som DHCP-tjeneren foreslo. Utdelingen avsluttes med DHCP ACK fra tjeneren som bekrefter at du nå har fått tildelt disse parametrene. Disse tildeles for en begrenset periode (”lease time”), og må fornyes.

Etter DHCP-utvekslingen over vil vi se:

```
C:\>ipconfig

Windows IP-konfigurasjon

Ethernet-kort eth0:

        Tilkoblingsspesifikt DNS-suffiks : nith.no
        IP-adresse . . . . . . . . . . . : 10.21.6.149
        Nettverksmaske . . . . . . . . . : 255.255.252.0
        Standard gateway . . . . . . . . : 10.21.4.1
```

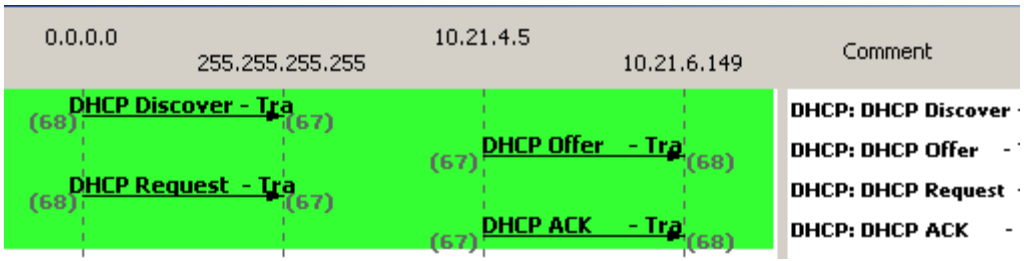
Figur 34: Nye nettverksparametre tildelt med DHCP

Dersom man ikke oppnår kontakt med noen DHCP-tjener vil man typisk ende opp med et oppsett som i figuren under:

```
C:\>ipconfig

Windows IP-konfigurasjon

Ethernet-kort eth0:

        Tilkoblingsspesifikt DNS-suffiks : nith.no
        Automatisk konfigurasjon av IP-adresse. . . : 169.254.145.229
        Nettverksmaske . . . . . . . . . : 255.255.0.0
        Standard gateway . . . . . . . . :
```

Figur 35: Selvkonfigurert IP, ingen gateway

Gangen i tildelingen er dermed som i figuren under:



Figur 36: Flyt i tildeling av nettverksparametre (Wireshark)

I eksempelet over ser vi at vi har blitt tildelt IP-adresse 10.21.6.149/22. Som vi har sett så er ikke dette en adresse som kan brukes til routing i Internett: den er ikke route-bar. Slike adresser betyr vanligvis at vi befinner oss på innsiden av en NAT-tjener.

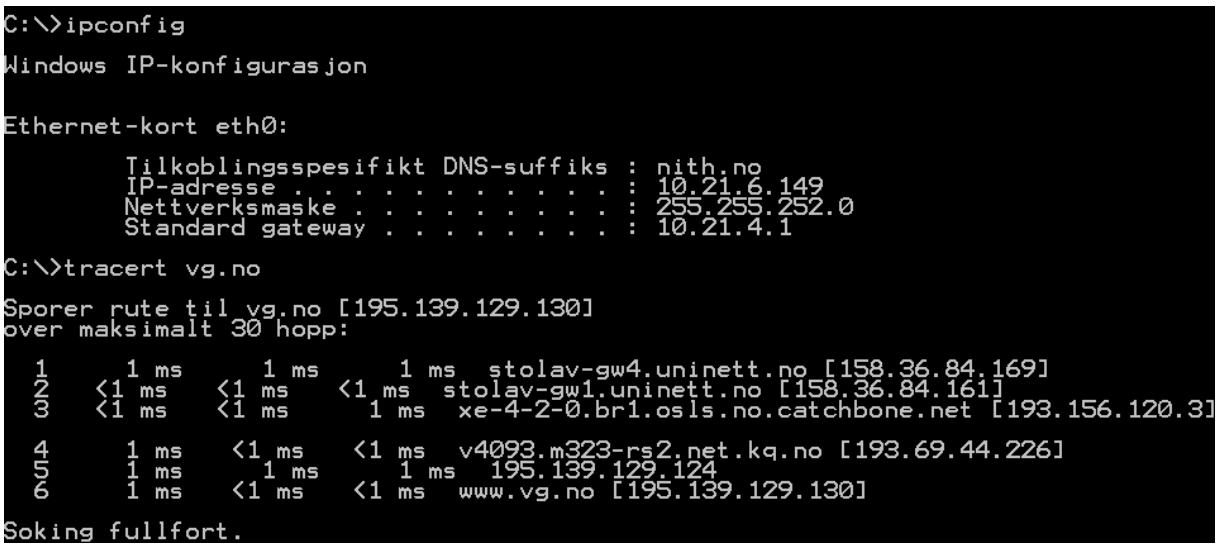
NAT: Network Address Translation

En IPv4-adresse er på 32 bit. Det tilsier at det det finnes ca 4 milliarder enkeltadresser. Så er det slutt. Pr september 2009 er siste overslag at alle IPv4 adresser vil være brukt opp i 2011 (se http://www.potaroo.net/tools/ipv4/index.html).

Årsaken til at det ikke har gått tomt for lengst er at de fleste ISPer, lokalnett og hjemmenett ikke bruker en IP-adresse pr maskin. I stedet bruker man

lokale/private adresser innenfor lokalnettverkene, og så er det bare routeren ut mot Internett (gatewayen) som har en reell (route-bar) IP-adresse.

I skjermbildet under ser vi at den lokale adressen er 10.21.6.149, men neste hopp er ikke 10.21.4.1 slik vi skulle vente. Dette er fordi på den routeren befinner det seg en NAT-tjener som erstatter den lokale 10'er-adressen medl en offentlig rout-bar adresse (158.36.84.169).



Figur 37: Tracert via NAT-server

NAT-tjeneren gjemmer altså et helt IP-nettverk bak (typisk) en enkelt adresse. For å kunne skille mellom de enkelte lokale host-adressene bytter den ikke bare ut avsender-adressen i IP-headeren. Den bytter også ut avsenders portnummer i transportlags-headeren (UDP eller TCP) og oppbevarer disse verdiene i en tabell den kan slå opp i når svaret kommer tilbake. Dette er antydet i tabellen under.

LOKALT		EKSTERNT	
Port-nummer	IP-adresse	Portnummer	IP-adresse
1254	10.21.6.149	1025	158.36.84.169
19875	10.21.4.131	1026	158.36.84.169
1232	10.21.5.101	1027	158.36.84.169

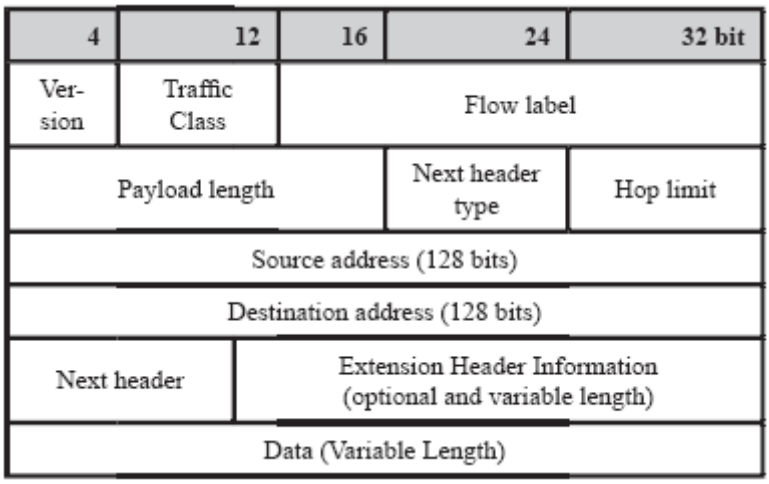
Når svaret kommer tilbake fra for eksempel en http-tjener så vil NAT-tjeneren slå opp i denne tabellen og sette inn igjen de opprinnelige verdiene, som så kan leveres til riktig maskin/adapter og riktig prosess/program på maskinen.

Denne "løsningen" på problemet med for få adresser bryter med ett av grunnprinsippene i TCP/IP-modellen, nemlig at underliggende lag skal yte tjenester for laget over. Her er det motsatt. Transportlaget brukes for å yte en tjeneste for nettverkslaget.

IP v6

På sikt skal problemet med for få IPv4-adresser løses ved overgang til IP versjon 6. Der er adressene på 128 bit. Det betyr at det finnes 79 milliarder milliarder milliarder (oktillioner) så mange adressers som det gjorde for IPv4.

IPv6 har en enklere og mer strømlinjeformet header enn hva IPv4 hadde:



Figur 38: IPv6 header

Noen av de viktigste forskjellene mellom IPv4 og IPv6 er:

- Sjekksummen og TTL feltene er fjernet. Routerene trenger ikke lenger å bruke kapasitet på disse.
- Broadcast er erstattet med flere ulike former for multicast
- Det skal med IPv6 være lettere å autokonfigurere host'er og routere.
- IPv6 skal ved "Flow Label" gjøre det enklere å sette opp routerne til å prioritere ut fra type trafikk. For eksempel streaming og Ip-telefoni *kan* behandles annerledes en bittorrent.
- Fragmentering er ikke lenger støttet/mulig i IPv6
- Støtter kryptering (IPSec) mer sømløst enn IPv4 gjorde.
- IPv6 skal gjøre det lettere å bytte IP-adresse automatisk når man er på farten (mobile IP)

Selv om moderne routere, DNS (AAAA records), nettverkskort, maskiner, operativsystemer, brukerprogrammer m.m. stort sett støtter IPv6 det svært få ISPer gjør det. Både fra og med Windows Vista og OS X så kommer de fleste maskiner med IPv6 klart til bruk (default påslått). De tilbyr da det som kalles en dual-stack løsning der man kun bruker IPv6 dersom mottager også benytter IPv6.

Hovedårsaken til at at ISPer sjelden støtter IPv6 er sannsynligvis fordi de har lagt så mye arbeid ned i IPv4 infrastrukturen og det er der de har sin kompetanse. Inntil de blir tvunget over, holder de seg til det kjente.

Dersom man benytter IPv6 i et bedriftsnettverk og bedriftens ISP tilbyr IPv6 er man likevel ikke garantert IPv6 forbindelse hele veien til alle mulige mottagere. Dette medfører at både lokalt, hos IPS og andre steder i Internett så vil IPv6-datagrammene måtte pakkes inn i et IPv4-datagram (tunneling) på veien.

IP v6 adresser noteres med CIDR-notasjon og hexadesimale sifre. En IPv6 adresse kan da se ut som følger: 2001:0db8:85a3:0000:0000:8a2e:0370:7334

Typer routing protokoller: LS vs DV

Routing-protokollene som brukes i nettverk kjøres av routerene og benyttes for å utveksle informasjonen de trenger for å sette opp sine respektive routing-tabeller. Tabellene settes opp for å finne hva som er billigste vei til et gitt nettverk. Hva billigst betyr avhenger av hvilken *metrikk* som routing-protokollen benytter. I det enkleste tilfellet kan dette være antall hopp til en mottager. Mer kompliserte metrikker vil ta hensyn til båndbredde/bitrate, router-tilgjenglighet, metning og lignende. På øverste nivå vil man også ta hensyn til pris på en gitt link, hvem man har gode peering-avtaler med, nasjonalt lovverk o.l. (policy routing).

Vi skiller mellom to hoved-*typer* routing protokoller:

- Link State. Her kringkaster alle routere (noder) i nettverket sine kostnader til alle tilkoplede naboer. Dersom det er 30 routere i nettverket så sendes da denne informasjonen til alle de 29 andre. Så benyttes en algoritme (f.eks. Dijkstras – se læreboken) til å beregne en routing-tabell som gir lavest kostnad fra seg selv til alle andre routere (og deres tilknyttede nettverk).
- Distance Vector. Disse protokollene er distribuerte, hver router kommuniserer bare med sine direkte tilknyttede naboer. Man kan si at oppdatering her foregår ved ”sladder”. Dersom en router får en billigere vei til en av sine naboer så sender den en melding til sine andre naboer om disse. De sjekker om de da vil ha en billigere vei til, eller via, denne. Dersom så ikke er tilfelle forblir routing-tabellene som før. I motsatt fall sender de oppdatert informasjon til sine egne nærmeste naboer.

Link State involverer dermed potensielt mye trafikk fordi hver router kommuniserer med alle. Distance Vector medfører vanligvis langt mindre trafikk, men er sårbar for andre mulige feilsituasjoner.

Som nevnt så benyttes routing-protokollen BGP mellom ISPer (Autonome Systemer). Innenfor bedriftsnettverk og ISPer benyttes gjerne protokoller som RIP (Distance Vector), OSPF (Link State), IS-IS (Link State), EIGRP (Distace Vector). Forskjellen mellom disse og virkemåten er det bare nattverksansvarlige som må vite noe om.

BGP-routerne som skal kunne route gjennom hele Internett må i september 2009 forholde seg til ca 30000 forskjellige AS-nummer og ca 300000 IPv4-nettverksprefix (se http://bgp.potaroo.net/). Det gir seg selv at det ikke er aktuelt benytte en Link State algoritme på (minimum) og mellom 30000 ulike routere. DA ville sannsynligvis ikke Internett vært brukt til stort annet enn å oppdatere routing-tabeller.

Kontrollspørsmål til kapittel 4

17. Hva er en IP-adresse? (Svar med en setning)
18. Hva er en nettmaske, og hva brukes den til? (Svar med to setninger)
19. Hva brukes default gateway til? Hvordan kan denne settes med route-kommandoen i WinXP?
20. Hvordan går du frem for å få se hva som er IP-adresse, nettmaske og default gateway for en arbeidsstasjon/PC?
21. Konverter (den desimale) IP-adressen 127.0.0.1 til binært format.
22. Konverter den siste oktetten i IP-adressen 10.21.4.80 til binært format.
23. Konverter den tredje oktetten i nettmasken 255.255.252.0 til binært format.
24. Hva er et broadcast-domene? Hvorfor er dette viktig?
25. Hvor mange noder er det plass til i et nettverk med nettmaske 255.255.255.224?
26. Hva er galt med nettmasken 255.255.255.208?
27. Hva er CIDR? Hvordan skiller dette seg fra "klassefull" (A,B,C) adressering?
28. Forklar hva notasjonen /25 og /24 angir og hvordan denne henger sammen med 255.255.255.128 og 255.255.255.0.
29. Hva er den laveste og høyeste adressen som kan brukes på en node i IP-nettverket 10.21.4.0 /22
30. Hva er broadcast-adressen i IP-nettverket 10.21.4.0/22?
31. En node har på CIDR-format adressen 10.21.11.135/19 Hva er adressen til IP-nettverket den tilhører (nettverksprefixen)?
32. Hva menes med subnetting? Hvorfor brukes subnetting?
33. Hva er aggregering av prefixer og hvordan gjøres dette i forbindelse med IP-routing?
34. Nevn 5 viktige forskjeller på IPv4 og IPv6
35. Hvor brukes routingprotokollen BGP?
36. Hva er et AS?
37. Hvordan skaffer man seg IP-adresser og AS-nummer? (Hint: RIPE.)

Kapittel: *Linklaget og lokalnettverk (LAN)*

Nettverkslaget i TCP/IP-modellen har med å flytte rammer (”frames”) mellom nabo-noder i nettverket å gjøre. På fysisk lag (som vi ikke behandler nærmere i dette kurset) så er oppgaven å finne ut hvordan enkelt-bit skal kodes og overføres ved hjelp av et elektromagnetisk signal. Link-laget handler da om hvordan d
I veiledningen vil vi i denne delen, som i de tidligere, ta utgangspunkt i hvordan link-laget benyttes pr dags dato. I praksis vil det si at vi ser på trådbunnet og trådløst Ethernet og legger vekten på de prinsippene og teknikkene som er viktige for å forstå disse teknologiene.

Link-lagets oppgaver og tjenester

Linklaget har som hovedoppgave å sørge for forbindelsen mellom naboer i nettverket. Dette involverer:

- Mux/demux: å ha en måte å angi hvilken protokoll på laget over i TCP/IP-stacken som skal benyttes
- Innramming: Å tilby en standard for hvordan datagrammer rammes inn i en header og trailer. Protokoll dataenheten (PDU) på linklaget kalles en ramme (”frame”).
- Pålitelig levering: Å tilby ulike grader av feil-sjekking og –behandling. Samt å tilby mekanismer for flytkontroll. Dette foregår i stor grad etter prinsipper tilsvarende de vi kjenner fra TCP.
- Medium Access Control (MAC): Å angi reglene for hvordan rammen skal sendes ut på overføringsmediet og hvem som skal kunne sende når.
- Full eller halv duplex: Skal man kunne sende samtidig som man mottar eller ikke?

Link-laget sine oppgaver ser dermed ikke ut til å være radikalt forskjellige fra oppgavene som løses på transport- og nettverks-laget. Det er likevel en viktig forskjell. Protokollene på transport- og nettverks-laget er de samme i hele Internett. Link-lag protokollene står det fritt å velge i hvert enkelt lokalnettverk (LAN), og på hver enkelt forbindelse. Man kan dermed ikke forutsette noe om hvilke tjenester som faktisk er implementert.

Dersom det er en ren punkt-til-punkt forbindelse slik man har med f.eks. bruk av et modem, *kan* linklagsprotokollene være svært enkle. PPP er et eksempel på en slik protkoll der man ikke trenger noen adressering av avsender/mottager på linklaget fordi det bare er en avsender og en mottager involvert. I lokalnettverk der flere skal sende og motta samtidig trenger man en mer komplisert løsning.

IEEE 802.3 (Ethernet)

Ethernet er sannsynligvis den mest brukte av alle link-lag protokoller. Dersom vi ser på den i Wireshark så får vi frem:

```
⊟ Ethernet II, Src: 00:22:55:3e:da:ba (00:22:55:3e:da:ba), Dst: 00:13:72:94:ff:78 (00:13:72:94:ff:78)
  ⊟ Destination: 00:13:72:94:ff:78 (00:13:72:94:ff:78)
      Address: 00:13:72:94:ff:78 (00:13:72:94:ff:78)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
  ⊞ Source: 00:22:55:3e:da:ba (00:22:55:3e:da:ba)
    Type: IP (0x0800)
    Trailer: 000000000000
```

Figur 39: Ethernet-header i Wireshark

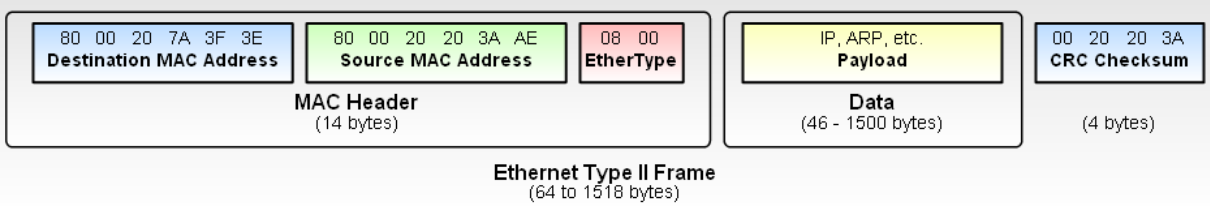
Ethernet-kort er utstyrt med MAC-adresser. Disse er på 48 bit (6 byte) og noteres med hexadesimale sifre. Mottageradressen over: 00:22:55:3E:DA:BA, betyr dermed:
00000000 0010 0010 01010101 00111110 11011010 10111010
i binært.

Adresse-rommet administreres av IEEE og de 3 første bytene tildeles produsenter etter søknad. De tre siste disponerer produsenten fritt til sine kort. 00:13:72 kan vi dermed slå opp og få vite at produsenten av kortet er Cisco (http://standards.ieee.org/regauth/oui/index.shtml).

MAC-adressen FF:FF:FF:FF:FF:FF benyttes til kringkasting (”broadcast”), og oppfattes da som adressert til alle i et LAN. I tillegg har de to siste bitene i første byte spesiell betydning. Dersom den siste er satt til 1 så skal rammen behandles om multicast, i praksis blir det nesten alle steder behandlet som kringkasting. Den nest siste byten skal være satt til 1 dersom man velger å benytte en annen MAC-adresse enn den som ble satt på kortet på fabrukken (virtuell MAC-adresse).

I sin helhet er formatet på en vanlig IEEE 802.3 ramme:

- Preamble: 8 byte. Består av 31 ganger 10 og avsluttes med 11.
- Mottager MAC-adresse: 6 byte
- Avsender MAC-adrsse: 6 byte
- Type-felt: 2 byte (0x800 er IP, 0x806 er ARP, osv)
- Sjekksum: 4 byte (CRC-32)
- Etter hver ramme skal det være 12 byte-tider med stillhet før en ny ramme kan sendes ut på mediet.



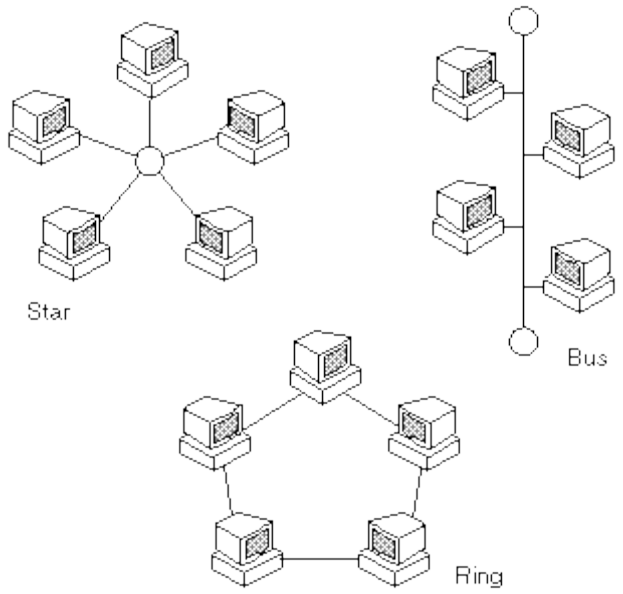
Figur 40: Ethernet type II header og trailer

Verken Preamble eller sjekksummen ser vi noe til Wireshark. Det er fordi Preamble-delen kun benyttes av nettverkskortene selv til å synkronisere klokkefrekvensen på mottager med avsender. Kortene fjerner denne delen før de viderformidler pakken. Vanligvis fjerner driveren til kortet sjekksummen før den leverer resten videre.

Sjekksummen i 802.3 brukes bare til å oppdage feil, ikke til å rett dem. Dersom det oppdages feil så droppes rammen uten tilbakemelding til avsender.

Det finnes mange forskjellige IEEE 802.3 varianter som støtter ulike bitrater og ulike tilleggstjenester (se http://en.wikipedia.org/wiki/IEEE_802.3). De vanligste i vår dager er 10Base-T, 100Base-T og 1000Base-T. Alle disse benytter uskjermede tvunnet trådpar (UTP) kabling og RJ-45 plugger og støpsel.

LAN kan ha ulike topologier: buss, stjerne og ring, avhengig av hvordan de er koplet sammen.



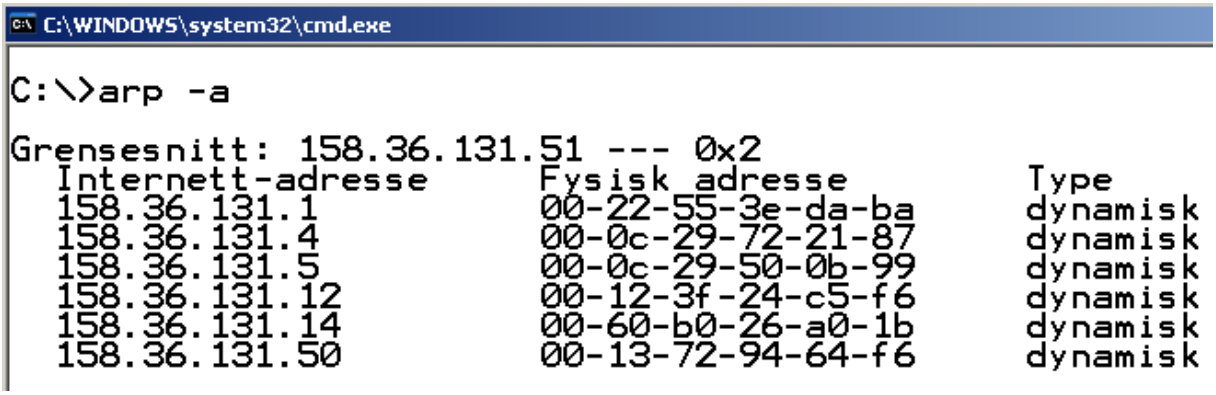
Figur 41: Nettverkstopologier: buss, stjerne og ring.

Kablet Ethernet er opprinnelig designet for å være et buss-nettverk. Designprinsippet for Ethernet kalles Carrier Sense Multiple Access / Collision Detection: CSMA/CD. Det betyr at alle kortene er koplet til samme buss (kabel), lytter på linjen, og kun sender dersom ingen andre gjør det. Dette er en variant av statistisk multiplexing (Se del 1). Siden det er en fysisk avstand mellom kortene vil kollisjoner oppstå når to kort starter å sende innenfor det tidsrommet det tar det elektriske signalet å nå det andre kortet. I henhold til IEEE 802.3 protokollen tillates kollisjoner og de behandles ved at begge kortene venter hver sin "tilfeldig" valgte periode før de forsøker å sende på nytt.

Sjekksummen som brukes av IEEE 802.3 er av typen CRC-32. Denne sjekksummen er mye bedre enn den som brukes på nettverks- og transport-lagsnivå. Bl.a. vil den oppdage alle feil som involverer 32 eller færre bits, alle feil som involverer et oddetall bits m.m.

Address Resolution Protocol (ARP)

Vi har sett at når TCP/IP-stacken i operativsystemet lager et datagram så adresseres dette med en IP-adresse. I et lokalnett der man bruker Ethernet og IP trengs dermed en mekanisme for finne ut hvilken Ethernet-adresse som hører sammen med hvilken IP-adresse. Dette gjør ARP. På hver maskin oppbevares en ARP cache, som kopler MAC-adresse til IP-adresse:



Figur 42: Utskrift av en ARP cache med ARP-kommandoen

Det er maskinene i nettverket som selv opplyser om sin IP-adresse når den får en ARP-forespørsel. ARP-forespørsler adresseres med FF:FF:FF:FF:FF:FF (kringkastes).

Nettverkskomponenter: Hub, Bro switch og router.

I starten var ethernet-nettverk bussnett der alle kortene i et segment av lokalnettet var koplet på samme kabel ved hjelp av vampyrklemmer. I våre dager er det typiske at de er koplet sammen ved hjelp av en aktiv switch. Dette medfører at den fysiske topologien i realiteten er et stjernenettverk.

En hub er en komponent på fysisk lag. Den tar i mot rammer som kommer inn på en port og sender dem videre på alle andre porter. I tillegg vil den vanligvis forsterke og "rense" signalet. Brukes ikke mye i våre dager.

En bro er en komponent på linklaget som kan oversette mellom forskjellige linklagsprotokoller. Denne funksjonaliteten er i våre dager stort sett bare nødvendig ved overgang fra kablet til trådløst nettverk, og på uplink-porten på switcher, der vi skifter mellom ulike versjoner av Ethernet.

En switch er en komponent på linklaget. Switcher bygger opp switching-tabeller med utgangspunkt i MAC-adressene den leser i rammene den mottar og kan da sende Ethernet-rammen ut på den porten som tilhører en bestemt MAC-adresse. Switchen bygger automatisk opp denne tabellen. Den lærer hvilke nettverkskort som er tilkoplet og bruker dette til å bygge opp en forwarding-tabell.

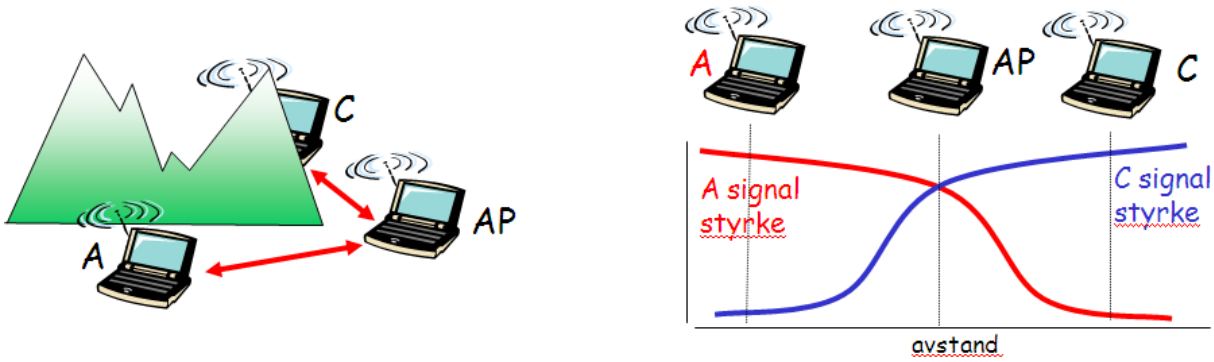
En router er en komponent på nettverkslaget, men vil oftest også inneholde switch- og bro-funksjonalitet. Hovedoppgaven til routeren er å kjøre routingprotokoller og bygge opp routingtabellen.

En typisk hjemme-router inneholder gjerne en 4 ports switch, broer mellom IEEE 802.3 og 802.11, har en uplink (WAN) port som leder videre til et modem (ADSL-, kabel, fiber) og inneholder gjerne også en NAT- og DHCP-server. Typisk fyller den også rollen som trådløst Access Point (AP). Det den sjelden gjør er å kjøre en routing-protokoll. Strengt tatt er den dermed nesten alt annet enn en router.

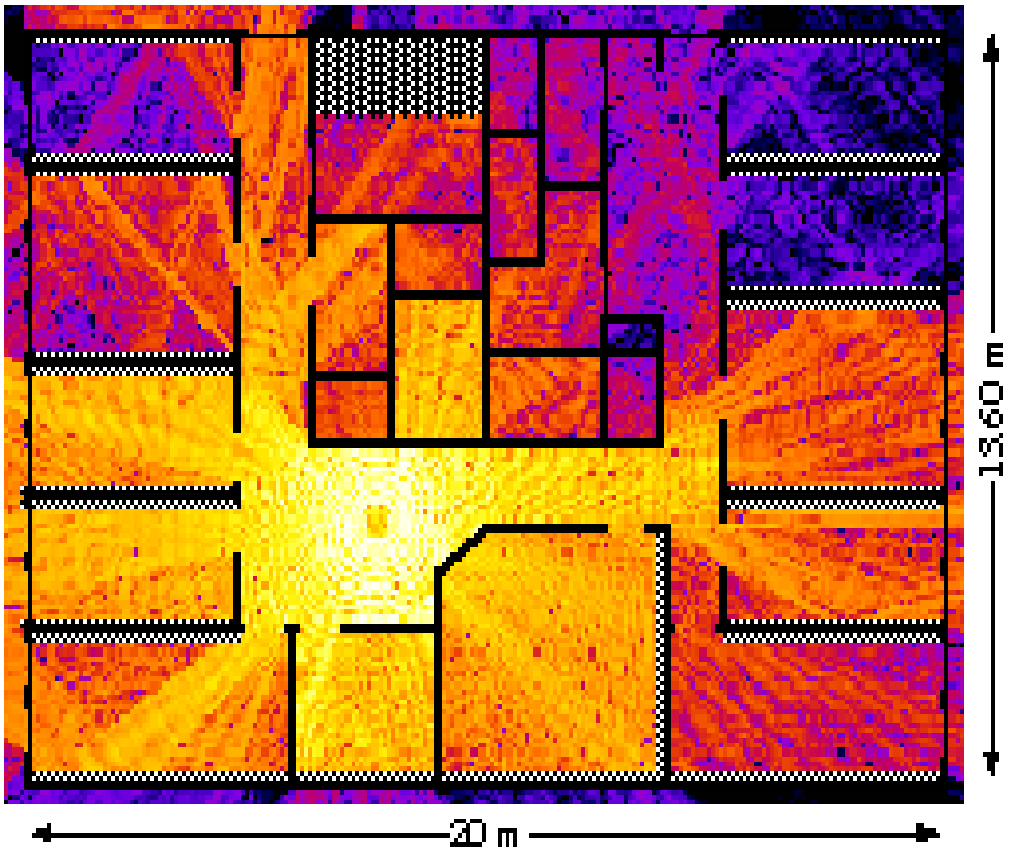
Trådløse nettverk (IEEE 802.11)

IEEE 802.11 serien av protokoller angir hvordan trådløse nettverk kan bygges opp. Fordi disse er basert på radio-overføring er det en rekke problemer som dukker opp i trådløse nett som vi vanligvis ikke vil oppleve i kablede:

- Mobilitet: brukeren er typisk en bærbar PC eller håndtholdt terminal som skifter posisjon.
- Støy: Det er langt flere mulige støykilder, og de er vanskeligere å forutsi og skjerme seg mot.
- Signalet sendes ut av en antenne. Det vil derfor synke i styrke med kvadratet av avstanden. Det vil også kunne avbøyes og reflekteres, noe som medfører at det samme signalet kan brukes ulik tid på nå frem til mottagerantennen.
- ”Hidden terminal”. Dersom man bruker et Access Point (AP) så vil to forskjellige terminaler kunne være i kontakt med det samme punktet, men ikke være i stand til å motta hvernadres signaler. Enten fordi det er fysisk hindring mellom dem, eller fordi avstanden mellom de to terminalene er for stor.



Figur 43: Hidden terminal problemet



Figur 44: Signalforplantning fra et sentralt Access Point

WiFi Alliance fungerer som standardiserings- og test-organisasjon for trådløse datanettverk.

Trådløse nettverk finnes i flere ulike utgaver a, b, g og n. De viktigste forskjellene mellom disse er oppsummert i tabellen under.

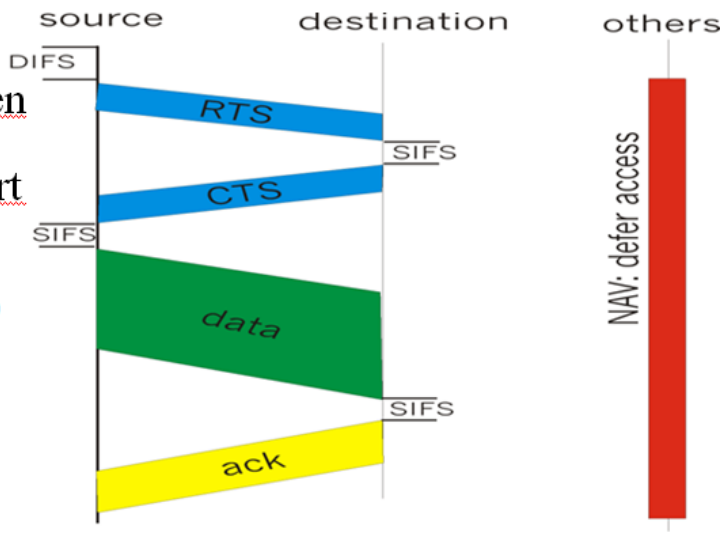
Type	Frekvensområde	Teoretisk max	Rekkevidde (innendørs)	Kommentar
802.11 a	5 GHz	54 Mbps	15 m	
802.11 b	2,4 GHz	11 Mbps	45 m	
802.11 g	5 GHz	54 Mbps	45 m	
802.11 n	2,4 eller 5 GHz	600 Mbps	91 m	Flere antenner

Frekvensområder tildeles av nasjonale myndigheter og de ”beste” områdene for radiotrafikk var allerede tildelt andre formål da trådløse datanettverk ble introdusert. Pga støy og andre forhold vil den teoretisk maximale bitraten aldri bli oppnådd i virkelige nettverk.

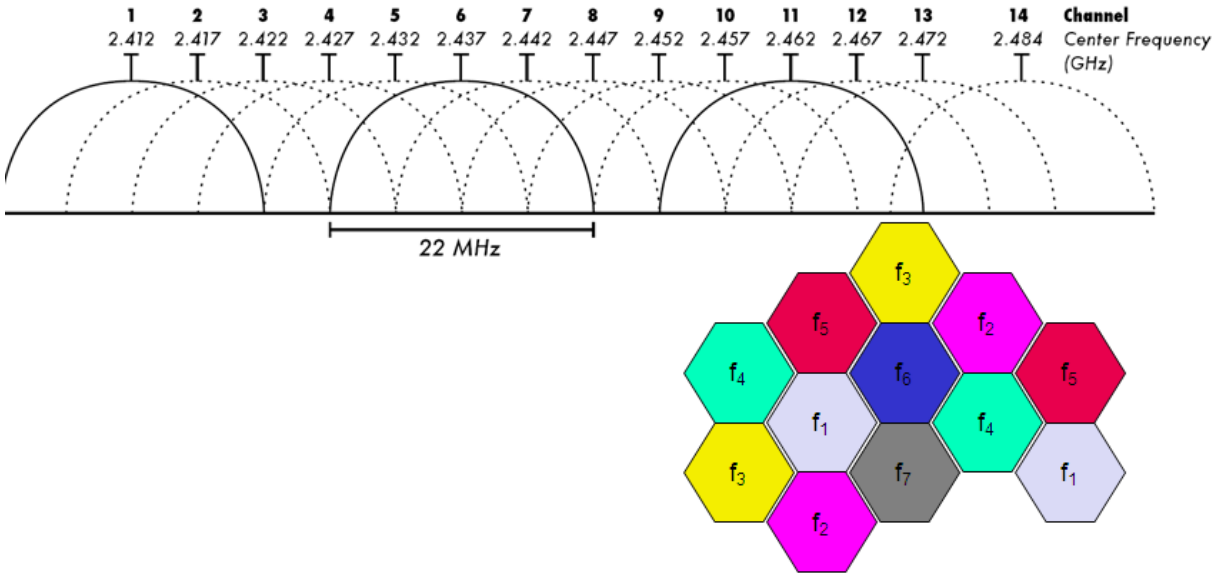
802.11 kan brukes i ”Ad Hoc” modus for å forbinde to terminaler ”på sparket”. F.eks. for å overføre filer fra en bærbar PC til en annen. Den vanligste bruksmåten er dog mot et sentralt Access Point. Hver celle i nettverket kalles da et BSS (Basic Service Set) og er identifisert med en SSID, som brukes ved pålogging og til å identifisere hvilke radiosignaler som tilhører ett enkelt nettverk.

Pga ”hidden terminal” problemet kan ikke trådløse nettverk bruke CSMA/CD. Det finnes ikke noen garantert måte å oppdage kollisjoner på. Derfor følger 802.11 CSMA/CA – Collison Avoidance. Det er flere mekanismer som skal forhindre kollisjoner. Den viktigste er at nettverkskortene kan reservere sendetid med en spesiell type RTS-rammer. Se figuren under:

- Sender overfører kort RTS (request to send) pakke: antyder varigheten på overføringen
- Mottager svarer med kort CTS (clear to send) pakke
  - varsler (muligvis skjulte) andre noder
- Skjulte noder avstår da fra å sende i det reserverte tidsrommet: NAV



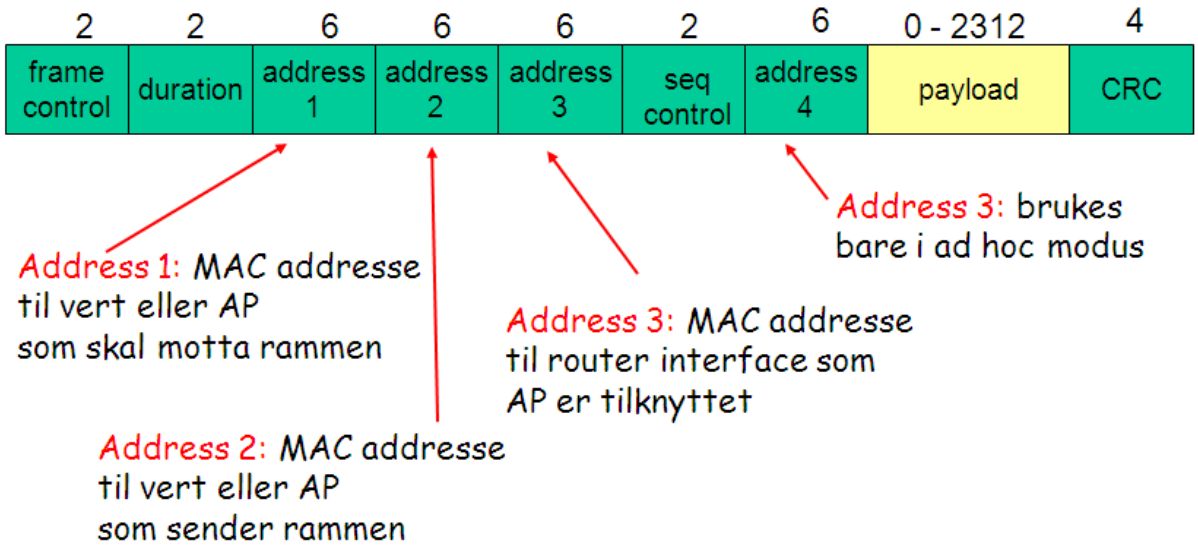
Figur 45: Reservasjon av sendetid



Figur 46: Oppdelingen av frekvensområdet i kanaler

Frekvensområdene er delt opp i kanaler. Access Point som er naboer skal benytte ulike kanaler. Kanalene er delvis overlappende, så i et nabolag vil man fortrinnsvis benytte kanaler som ligger lengst mulig fra hverandre.

Rammene i 802.11 er også forskjellige fra 802.3. For det første så er de større og kan flytte mer data (2312 byte; 812 byte enn max i 802.3). For det andre har de en mer komplisert header. For eksempel så er det plass til hele fire forskjellige MAC-adresser.



Figur 47: 802.11 ramme.

Øvingsoppgaver Nettverks- og linklaget

Øvingsoppgave 1

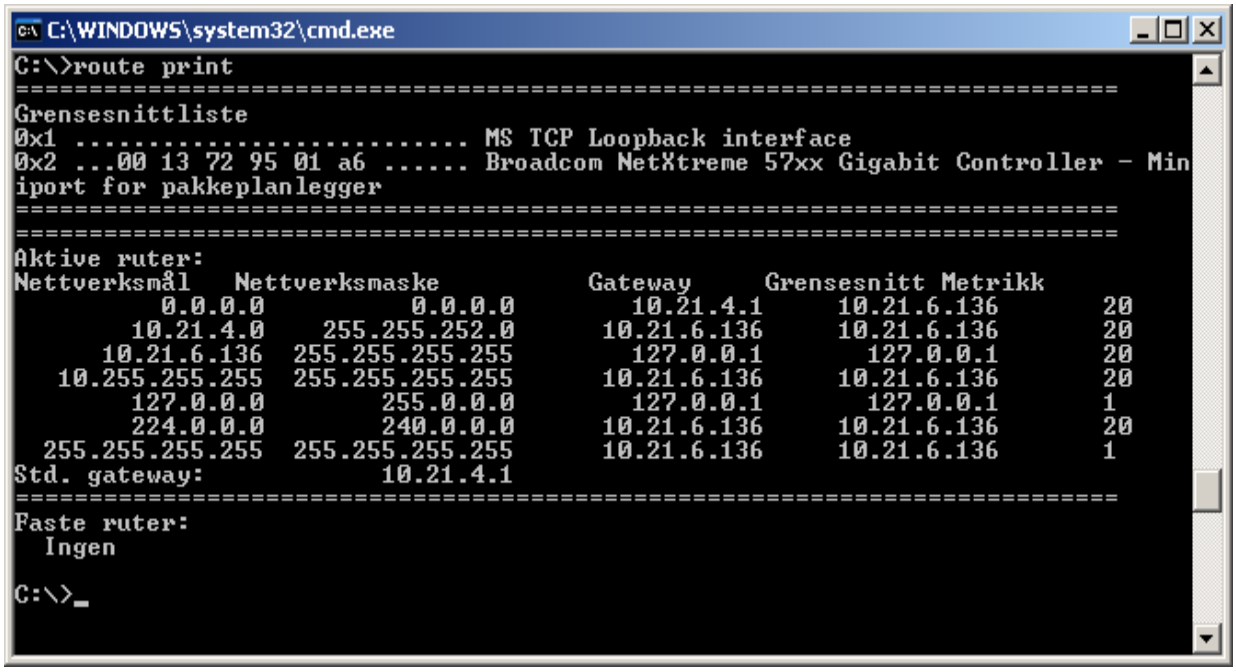
a. Bruk ipconfig til å få en oversikt over nettverksparametrene som er satt for adapterene på din egen maskin.

b. Bruk arp-kommandoen til å få en oversikt over hva som ligger i arp-cachen på din egen maskin.

c. Bruk route print (netstat –r) til å liste opp din lokale routing-tabell og tolk innholdet.

Øvingsoppgave 2

Med kommandoen `route print` kan du få se hvordan forwarding-tabellen i maskinen du jobber på ser ut:



Figur 48: exempel på route print

Bruk denne kommandoen på maskinen du arbeider på.
Ta utgangspunkt i tabellen under og fyll inn de binære utgavene av IP-adressene til "Nettverksmål" og "Nettverkmaske" for de verdiene du får opp på *din maskin:*
.

Nettverksmål		Nettverksmaske	
Desimal	**Binær**	**Desimal**	**Binær**
0.0.0.0	0000 0000 . 0000 0000 . 0000 0000 . 0000 0000	0.0.0.0	0000 0000 . 0000 0000 . 0000 0000 . 0000 0000
10.21.4.0	0000 1010 . 0001 0101 . 0000 0100 . 0000 0000	255.255.252.0	1111 1111 . 1111 1111 . 1111 1100 . 0000 0000

For å avgjøre hvilket interface som en IP-pakke skal sendes ut på benyttes "prefix matching". (Se læreboken)
Dvs at der nettmasken til en IP-adresser er lik 1 er prefix-delen av den samlede adressen.
Tabellen over ser dere at er sortert fra laveste til høyeste nummeriske verdi. Det sier seg selv at dette ikke kan være den rekkefølgen som maskinen benytter nå den skal velge hvilket interface en bestemt IP-pakker skal sendes ut på. Da starter den med de lengste prefixene og fortsetter nedover forwearding-tabellen til den finner en match. 0.0.0.0 med nettmaske 0.0.0.0 ligger dermed alltid på bunnen av tabellen.

Ta utgangspunkt i tabellen du har laget over, og fyll inn tabellen under i riktig rekkefølge med prefixene du finner når du kun legger inn de bitsene av IP-adressene som nettmasken dekker. Fyll også inn hvilket interface som skal benyttes for hvert prefix.

Prefix	Interface (Grensesnitt)
1111 1111 . 1111 1111 . 1111 1111 . 1111 1111	2
"Alt annet" mao 0.0.0.0 m/ nettmaske 0.0.0.0	2

`route` kommandoen kan også benyttes til å legge inn statiske ruter i forwardingtabellen. Kan du forstille deg noen sikkerhetsmessige problemer med dette?

Øvingsoppgave 3

Bruk tracert til å observere hvilken vei som velges fra din hjemmemaskin og til www.vg.no. (Dersom du ikke får opp annet enn * på hvert trinn kan dette ha flere årsaker. Du har satt opp enn brannmur som stopper ICMP-trafikk, den må du eventuelt slå av. ISPen kan ha en brannmur som stopper ICMP-trafikk, da er det lite du kan gjøre.)

a. Det finnes mange forskjellige mer "brukervennlige" utgaver av tracert. To av dem, som er verdt å sjekke ut er pingplotter (http://www.pingplotter.com) og VisualRoute (http://www.visualware.com/) Test ut disse mot en del adresser du bruker ofte.

b. På nettstedet http://traceroute.org finner du en liste over ”nettsteder” rundt om kring i verden som tilbyr deg å kjøre traceroute derfra.
Velg ut fire steder (fortrinnsvis så langt unna som mulig?) og kjør traceroute mot din eksterne IP-adresse. Du kan også bruke gw.nki.no (213.172.201.1).
Kjør selv traceroute mot disse serverene.
Følger pakkene samme ruter i begge retninger? Diskuter hvorfor/hvorfor ikke.

Øvingsoppgave 4

a. (Dersom du selv har satt en statisk IP-adresse kan du hoppe over denne øvingsoppgaven) Med kommandoen `ipconfig /release` kan gi opp den IP-adressen som DHCP-serveren din har tildelt deg. Med `ipconfig /renew` kan du be om å få tildelt nettverksparametrene på nytt av tjeneren.
Bruk Wireshark til å fange opp trafikken de to kommandoene utløser og forklar resultatet.

b. Forklar hvordan NAT virker.
Sannsynligheten for at du sitter bak en NAT-router er stor.
Hvordan vet du det dersom du sitter på et nettverk som ikke bruker NAT?
Bruk tracert til å kartlegge veien ut i Internett og finn ut hva som er den første "virkelige" IP-adressen.

Øvingsoppgave 5

a. I de fleste Ethernet-adresser kan man lese ut hvem som er produsenten av adapteret som rammen skriver seg fra av de tre første bytene i MAC-adressen. Bruk listen over Ethernetkortprodusenter som du finner sammen med Wireshark-installasjonen din (vanligvis plassert i filen: C:\Programfiler\Wireshark\manuf) til å finne ut hvem som har produsert et kort med MAC-adresse 00-13-72-94-FF-78
Prøv også å bruke kommandoen ipconfig /all og manuf-listen til å finne hvem som "eier" MAC-adressen på din egen maskin.

b. Figuren under viser de ulike feltene innenfor en Ethernet-ramme. Sett navn på disse, forklar hva de er til, og oppgi størrelsen i byte

Øvingsoppgave 6

a. Båndbredden i et Ethernet er oppgitt til 10 Mbps. Gjør et overslag på hvor lang tid tar det å sende ut en bit. Gjør det samme for et 100 Mbps nettverk og 1 Gbps netter.

b. Hva er den minste og største størrelsen en Ethernet-ramme kan ha?
Hvor lang tid tar det for et 100 Mps adapter å sende disse ut på linjen?

c. Du kan gå ut fra at signalet forplanter seg med ca 2/3 av lyshastigheten langsmed kabelen (nærmere bestemt 224 844 km/s).
Foreta et overslag over hvor mye plass (i meter) hver enkelt bit bruker når båndbredden er hhv 10 og 100 Mbps.
(Du har allerede funnet halve svaret i deloppgave a...)
Hvor lang i meter blir da den korteste og lengste rammen du kan sende i hhv et 10 og 100 Mbps Ethernet?

Øvingsoppgave 7

Ethernet II (IEEE 802.3) benytter CSMA/CD. Forklar hva dette innebærer, og hvordan kollisjoner håndteres.

Ethernet 802.11 standardene benytter ikke CD, men CA. Hvorfor det?

Forklar hvordan CA ivaretas i et 802.11 nettverk

Hva er de viktigste forskjellene på 802.11 a, b, g og n?