Kapittel 3

Anvendelser av Internett, applikasjonslaget

Læringsmål: Etter å ha lest dette kapitlet skal du

- forstå hvordan webtjenesten fungerer
- forstå hvordan e-posttjenesten fungerer
- forstå hvordan navnetjenesten fungerer
- skjønne hvordan protokoller styrer utveksling av meldinger mellom klient og tjener i Internett-applikasjoner

3.1 Hva er Internett?

Internett (med stor «I») er navnet på et bestemt datanettverk som egentlig er en sammenkopling av mange nettverk. Et kjennetegn ved Internett er at det bruker en bestemt protokollfamilie for datakommunikasjon, nemlig TCP/IP. I dagligtalen er Internett også synonymt med de tjenestene som man kan få utført, som e-post og web. Hele poenget med Internett er å kunne kommunisere mellom program som kjøres på ulike maskiner.

Internetts infrastruktur: Dette innebefatter nettverk og utstyr. Som brukere kopler vi datamaskiner av en eller annen utforming og størrelse til Internett, og dette kalles endeutstyr. Endeutstyret er samlet i større eller mindre bedriftsnett og private hjemmenett. Det er vanlig å si at Internett består av kjernenett og kantnett, og endeutstyret er i kantnettet. I kjernenettet er det kraftige rutere som formidler trafikken. Det finnes også utstyr som gir overganger mellom Internett og andre typer nett, for eksempel telefonnett og mobilnett, og slikt utstyr kalles for en portner (*gateway*). Kommunikasjonslinjer knytter utstyret sammen, og forbindelsene varierer fra vanlige kobberlinjer inn i de tusen hjem til fiberkabler mellom internasjonale knutepunkter.

Internetts tjenester: To av de mest populære anvendelsene av Internett er e-post og web. Dessuten er chatting svært populært, og IP-telefoni er i ferd med å ta av. Det var anslagsvis 80 000 kunder med IP-telefoni i Norge i mai 2005. Felles for alle anvendelsene er at det er program (applikasjoner) som kommuniserer med hverandre over Internett.

Internetts roller: Tilkopling til Internett skjer gjennom en ISP, en Internet Service Provider, altså en tjenesteleverandør. Tjenesten som leveres, er en tilgang til Internett til en avtalt hastighet og pris. Som oftest følger det mer med i abonnementet hos en ISP (e-postadresser, virusfilter, brannmur og mulighet for egne hjemmesider), men dette er ikke påkrevd for å kunne bruke Internett. Det du først og fremst trenger, er en IP-adresse, fordi alle tilknytninger til Internett må ha unike IP-adresser for å kommunisere med hverandre. Alle ISP-er har på forhånd fått tildelt (kjøpt) en blokk med IP-adresser fra større nasjonale eller internasjonale forvaltningsorganisasjoner.

Når man så skal bruke Internett, enten det er web eller e-post, bruker man domenenavn. I Norge er det NORID som forvalter domenenavn, det vil si har ansvar for «no-domenet».

Det å sende datapakker over Internett krever selvfølgelig at tolkning av datafelter og håndtering av innholdet er standardisert, det vil si at programmene oppfører seg likt på alt slags utstyr. Det sentrale standardiseringsorganet for Internett heter IETF (se avsnitt 2.3.4).

Internetts trafikk: Datatrafikken på Internett er basert på pakkesvitsjing. Det betyr at data samles i passende blokker som sendes hver for seg. Hver pakke får en merkelapp som vi kaller pakkeheader. Den viktigste informasjonen i pakkeheadere er mottaker- og avsenderadresse, slik at pakkene finner frem. De øvrige informa-

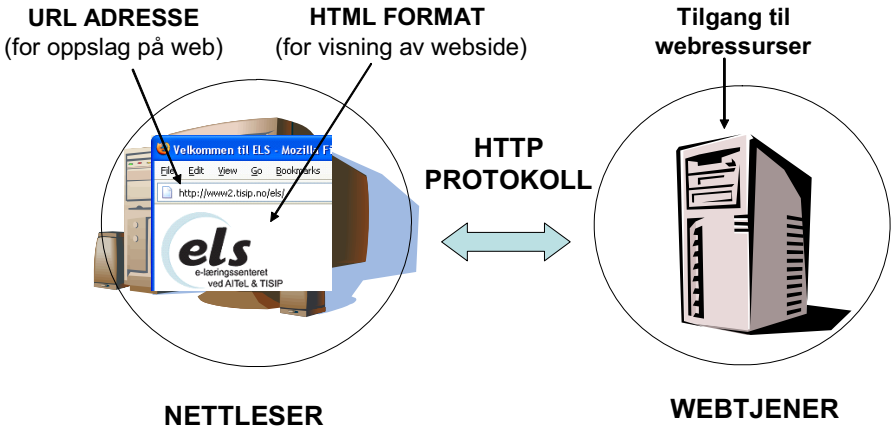
sjonsfeltene i pakkeheadere brukes for at kommunikasjonen skal håndteres på tilsiktet vis, for eksempel deresom det skulle oppstå feil i overføringen.

Internett generelt har ingen garantier for tjenestekvalitet. Det betyr at kapasiteten i nettet kan synke hvis det plutselig er mange som skal overføre mye data samtidig, og man opplever da treg respons. Bedrifter med spesielle krav til tjenestekvalitet kan kjøpe teletjenester som går utenom Internett.

Hele Internett med alle dets anvendelser bygger på den lagdelte kommunikasjonsmodellen. Vi starter gjennomgangen på toppen, det vil si nærmest brukeren.

3.2 Web

Web (verdensveven, World Wide Web) er noe av det man bruker Internett mest til. Når man «surfer» på nettet, bruker man HTTP-protokollen for utveksling av meldinger mellom partene. Partene i denne sammenhengen er klientprogrammet (nettleser, *browser*), som sender forespørsler om tilgang til webressurser, og webtjenere, som svarer. Webressursene adresseres på et format som kalles URL. HTML-koden beskriver hvordan websider skal presenteres på skjermen.



Figur 3.1 Hovedkomponenter som inngår i webtjenesten

3.2.1 Nettleseren

Klientprogrammet, eller bare klienten, for webtjenesten kalles nettleser. Eksempler på slike program er Microsoft Internet Explorer, Mozilla Firefox og Opera. Nettleseren sender en forespørsel til en webtjener om å få vist en webside og bruker HTTP-protokollen for å kommunisere. Nettleseren kan hente mange slags typer webressurser. Webressurser er et vidt begrep som omfatter alt som kan lagres elektronisk: websider, rene tekstdokumenter, bilder, lyd og video. Alle nettlesere

kan vise vanlige websider direkte, men for avspilling eller visning av enkelte andre typer ressurser må det installeres ekstra programtillegg (*plug-ins*).

Hovedfunksjonen til nettleseren er å vise webdokumenter i rett format på skjermen. Nettleseren må forstå hvordan den skal vise tekst og tabeller, hvilke fonter og farger som gjelder, plassering av bilde og så videre. HTML er et språk som beskriver et slikt format, og som nettleseren må kunne tolke.

Nettlesere har dessuten funksjoner for å kunne tilby et brukervennlig grensesnitt, som å lagre favorittadresser, skrive ut sider og så videre. I det hele tatt er nettlesere temmelig avansert programvare.

3.2.2 Webtjeneren

En *webtjener* står hele tiden og lytter etter forespørsler fra klienter. En forespørsel består som regel i å få lest et objekt. Webtjeneren leter frem det objektet som det spørres om, og sender det tilbake i en HTTP-svarmelding. Vi kan se på webtjenere som store fillagre som man leser fra, og filene kan ha forskjellig type innhold. En populær webtjener er Apache.

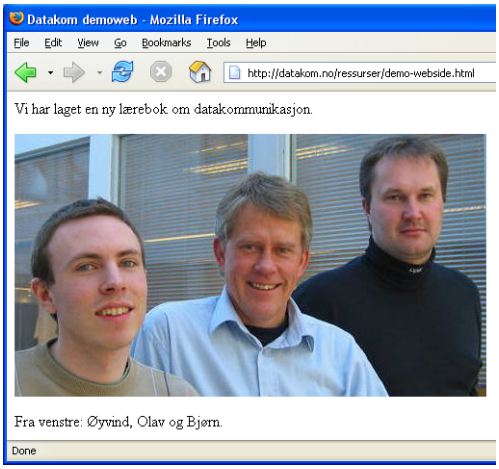
Når en ISP tilbyr sine kunder mulighet for å ha egne hjemmesider, betyr det at de stiller lagringsplass til rådighet for kunden på sin webtjener. Kunden må laste opp innholdet som skal vises.

Hovedstrømmen av data er fra webtjener til klient, men enkelte ganger sendes det data i motsatt retning, for eksempel når man har fylt ut et webskjema. Disse data blir mottatt og prosessert på tjenersiden, for eksempel registrere en kurspåmelding, foreta en banktransaksjon eller nær sagt hva som helst. Dette krever mer avansert webprogrammering enn konstruksjon av enkle hjemmesider.

3.2.3 HTML

HTML (HyperText Markup Language) er et programmeringsspråk for å konstruere websider. Det er ikke denne bokas oppgave å forklare HTML, men vi vil peke på hvordan websider kan bli konstruert ved å sette sammen flere objekter. Nettleseren får inn et HTML-dokument som beskriver hovedoppsettet av en webside. Nettleseren går gjennom koden og noterer seg en liste over alle objektene som refereres. Disse leses deretter inn i tur og orden og blir sammenstilt av nettleser for fremvisning som ett skjermbilde. Objektene trenger ikke engang å være lagret på samme webtjener.

Det er laget en enkel webside som brukes for demonstrasjoner i forbindelse med denne læreboka. Den har adressen http://datakom.no/ressurser/demo-webside.html og vises på neste side.



Figur 3.2 En enkel webside som brukes som eksempel i denne læreboka

Denne websiden er konstruert med HTML-kode. Hele denne websiden trenger ikke mer enn disse linjekodene:

Utlisting 3.1 HTML-koden til eksempel webside

```
1  <html><head><title>Datakom demoweb</title></head>
2  <body>
3
4  <p>Vi har laget en ny lærebok om datakommunikasjon.</p>
5  <p><img border="0" src="forfatterne.JPG" width="490"
6      height="273"></p>
7  <p>Fra venstre: Øyvind, Olav og Bjørn.</p>
8
9  </body>
10 </html>
```

HTML bruker det som kalles «tags», som skrives på formen <tag1> og </tag1> for å markere start og slutt på det som hører sammen i programlogikken. På websiden er det med et bilde av de tre glade forfatterne, men hvordan finner nettleseren dette bildet? Bildet er *referert til* i linje fem i kildeteksten, hvor det blant annet står `src="forfatterne.JPG"`. Dette er navnet på bildefila, og akkurat her er det brukt en såkalt relativ adressering. Det betyr at bildet ligger lagret i samme filmappe som HTML-dokumentet, derfor trengs ikke noen komplett URL. Relativ adressering gjør det enklere å kopiere og flytte hele mapper med webinnhold som hører sammen, uten å måtte oppdatere en rekke URL-referanser.

3.2.4 URL

URL (Uniform Resource Locator) er en form for adresse som viser hvor en ressurs er lagret. Ordet *locator* henspiller på at det gjelder å «lokalisere» ressursene vi skal hente på Internett. Man bruker URL-formen for eksempel i adressefeltet i nettleseren og i lenker som ligger på en webside. En URL skrives på et bestemt format, som ser slik ut:

URL-format:<*protokoll*>://<*domemenavn*><*:portnummer*>/<*ressursnavn*>.

En del av informasjonen i dette adresseformatet kan være underforstått, det vil si at det ikke må oppgis, det tolkes automatisk av nettleser og webtjener. Hvis man i nettleseren bare skriver webadressen datakom.no, så betyr det «bruk HTTP-protokollen og kople til maskinen som betjener domenenavnet *datakom.no* på port 80, og hent webdokumentet med navnet index». I dette tilfellet er både protokoll, portadresse, plassering i filarkivet og navn på webdokumentet underforstått.

Det er vanlig å bruke begrepet URL i forbindelse med webadresser, men egentlig inngår dette adresseformatet i det mer generelle begrepet URI (Uniform Resource Identifier). URI-formen er definert i RFC 3986.

3.2.5 HTTP

HTTP (HyperText Transfer Protocol) er protokollen som spesifiserer hvordan forespørsler og svar, det vil si *meldinger*, skal utveksles mellom klient og tjener i webtjenesten. HTTP er definert i RFC 1945 (versjon 1.0) og RFC 2616 (versjon 1.1). Meldingene er tekstbaserte. Det betyr at ledetekst og innhold er skrevet med vanlige tegn, noe som er relativt enkelt å tolke for de programmene som er involvert.

3.2.5.1 HTTP-forespørsel

En nettleser sender en forespørsel på et bestemt format når den ber om å få lastet ned et dokument. Forespørslen består av tre hovedelementer:

- en forespørselslinje
- diverse headerlinjer
- en kropp med eventuelle data

Dette kan eksemplifiseres ved å vise innholdet i forespørselen som nettleseren sender når den ber om å få bokas demonstrasjonswebside lastet ned. Den hentes fra http://datakom.no/ressurser/demo-webside.html. Pakken er analysert med Wireshark.

Figur 3.3 HTTP-forespørsel om en webside

Forespørselslinjen har formatet <*metode*> <*filnavn*> *HTTP/* <*versjon*>. Metoden GET, som vises i eksemplet, er den desidert vanligste og brukes når man ber om en fil fra webtjener. En annen metode er POST, som brukes i forbindelse med sending av data, for eksempel dersom man har fylt ut et skjema. HTTP definerer også flere metoder som kan benyttes i mer spesielle anledninger, disse finner man beskrevet i standarden.

Filnavn viser sti og dokumentnavn, som er */ressurser/demo-webside.html* i dette eksemplet. Dette alene er ikke nok for å lokalisere webressursen på maskinen man er tilkoplet, man trenger også domenenavnet. Domenenavnet hentes fra URL-en og sendes som tilleggsinformasjon i en headerlinje, eksempelvis «Host: datakom.no». Til sammen er dette tilstrekkelig informasjon for å be om en webressurs.

Versjon i forespørselslinjen er enten 1.0 eller 1.1. Versjon 1.1 er omtrent enerådende i dag, men systemet er bakoverkompatibelt så de kan benyttes om hverandre.

Headerlinjene varierer i antall og innhold avhengig av hvordan nettleseren er satt opp. Headerlinjene er omtalt fra side 100 og utover i RFC 2616 for HTTP versjon 1.1. Hensikten med headerlinjene er at webtjener skal kunne tilpasse svaret best mulig. For eksempel forteller «Accept-Language» hva slags språk brukeren foretrekker dersom webressursen finnes på flere språk. Vi skal se nærmere på bruken av «If-Modified-Since» og «Connection» i avsnitt 3.2.6.

Kroppen inneholder vanligvis ikke data ved GET-forespørsler. I de tilfellene hvor brukeren sender data, kommer datainnholdet her.

3.2.5.2 HTTP-svar

Webtjeneren sender svar på brukerens forespørsel etter beste evne. Svaret inneholder tre hovedelementer:

- en statuslinje
- diverse headerlinjer
- en kropp med eventuelle data

Vi skal nå studere svaret som webtjeneren ga på forespørselen om å laste ned lærebokas demonstrasjonswebside.

Figur 3.4 HTTP-svar fra webtjener

Statuslinjen har formatet *HTTP/<versjon> <statuskode> <statustekst>*. Vi ser at versjonen 1.1 brukes også på webtjeneren. Statuskoden er 200, og i klarteksten er det ok. Andre statuskoder man kan støte på, er 404 (forespurt objekt ikke funnet) og 401 (autorisasjon påkrevd). Merk at HTTP sender både statuskodenummer og klartekst, så man slipper å slå opp i referansebøker. Dersom man får feilkoder i svaret, kan driftsansvarlig for webtjeneren velge å sende ekstra headerlinjer med hint om hva som kan prøves.

Headerlinjene gir tilleggsinformasjon som klienten kan velge å benytte. Her vil vi kommentere «Keep-Alive» i underavsnitt 3.2.6.2.

Kroppen med data kommer etter headerlinjene dersom forespørselen resulterer i at man får et dokument tilbake. I dette eksemplet var det ingen data fordi forfatterens nettleser hadde en lokal kopi av dokumentet fra forrige gang det ble lest. Lokal mellomlagring omtales i underavsnitt 3.2.6.1

Merk at dersom et dokument er for stort til å overføres som svarmelding i én og samme pakke, må dette deles opp og sendes som flere pakker på nettet. Det kan være greit å huske på at dette fortsatt bare er én HTTP-svarmelding, selv om det kan se ut som mange HTTP-pakker når man bruker verktøy som Wireshark for å analysere trafikk på nettet.

3.2.6 Metoder for å effektivisere webtjenesten

Fordi web brukes så mye, er det viktig at HTTP jobber mest mulig effektivt. I den sammenheng omtales lokal mellomlagring, vedvarende forbindelser, pipelining, parallelle forbindelser og bruk av informasjonskapsler.

3.2.6.1 Lokal mellomlagring

Dersom nettleseren har mellomlagret kopier av webobjekter lokalt på maskinen (*caching*), kan disse kanskje brukes på nytt når vi leser samme webside flere gan-

ger. Forutsetningen for gjenbruk er at de ikke er endret siden sist på webtjeneren; dersom det finnes nyere versjoner, vil vi heller bruke dem. Lokal lagring er en funksjon som koster ekstra kompleksitet og lokal lagerplass. Gevinsten er raskere responstid i fremvisning av websiden, og man unngår å belaste nettet unødvendig.

Nettleser undersøker først om objektet som skal lastes ned, finnes lokalt lagret. Hvis så er tilfelle, sender nettleseren med en ekstra headerlinje i HTTP-forespørselen som sier når det ble opprettet. Webtjeneren undersøker om den har en nyere versjon av objektet. Hvis den har det, sendes hele fila som svar på forespørselen. Hvis webtjeneren ikke har noen nyere versjoner av objektet, sendes bare en statusmelding om at det ikke er endret, underforstått: Bruk det objektet du har.

Her er et eksempel på meldingsutveksling når nettleseren har funnet en kopi i lokalt mellomlager av det objektet som skal vises. Webtjeneren svarer at fila ikke er endret siden sist:

Klienten føyer til følgende headerlinje i GET-forespørselen:
`If-Modified-Since: Tue, 14 Jun 2005 13:56:09 GMT`

Webtjener svarer med følgende statuslinje i sin respons:
`HTTP/1.1 304 Not Modified`

Noen typer dokumenter lastes alltid ned på nytt selv om det finnes en lokal kopi. Dette gjelder dynamiske dokumenter, skrevet i for eksempel PHP eller ASP. HTML-kode genereres på sparket, slik at det alltid eksisterer en «nyere» versjon.

Lokal mellomlagring kan også ivaretas på en maskin som er felles for flere brukere.

3.2.6.2 Vedvarende forbindelser

Nettleseren går gjennom HTML-koden for en webside og lager seg en liste over alle objekter som skal lastes ned. I den opprinnelige utgaven av HTML (versjon 1.0) ble disse objektene lastet ned i tur og orden på den måten at klienten etablerte forbindelse med tjeneren, fikk et objekt overført og koplet forbindelsen ned igjen. Denne prosessen repeterte seg til alle objekter var overført. Nøyaktig hvordan slike forbindelser mellom klient og tjener håndteres, skal vi studere i forbindelse med TCP-protokollen, men inntil videre kan vi anta at opp- og nedkopling av forbindelser tar tid og gir ekstra belastning på maskin og nettverk.

Med HTTP versjon 1.1, som i all hovedsak brukes i dag, er det mulig å ha vedvarende forbindelser (eng. *persistent connections*). Det åpner for at flere objekter kan lastes ned over samme forbindelse. Målinger har vist at websider med mange små objekter kan lastes ned på bare halve tiden med denne metoden. Dette forutsetter selvsagt at de aktuelle objektene befinner seg på samme webtjener, ellers må det jo etableres nye forbindelser uansett.

Håndtering av vedvarende forbindelser styres av headerlinjene. Eksempel:

Klienten skriver: `Connection: Keep-Alive`

Webtjeneren svarer: `Keep-Alive: timeout 15, max=96`

Dette betyr at klienten ber webtjeneren om å holde forbindelsen oppe, og webtjeneren svarer «ja, det skal jeg gjøre», men den setter et tidsutløp. I eksemplet står det «timeout 15». Klienten har altså 15 sekunder på seg til å sende neste forespørsel om et objekt før forbindelsen stenges. Dette passer godt for vanlig surfing. Nå er det jo ingen krise om forbindelsen stenges; det kan åpnes nye, det tar bare litt lengre tid.

For ordens skyld nevnes at parameteren «max=96» betyr at du uansett ikke får lest flere enn 96 objekter på denne ene forbindelsen. Dette er et lite sikkerhetstiltak mot overbelastning av webtjeneren.

3.2.6.3 Vedvarende forbindelser med pipelining

HTTP versjon 1.1 tilbyr også muligheten for pipelining over en vedvarende forbindelse. Uten pipelining vil nettleseren be om ett objekt, få det vel og bra overført og så be om neste objekt på lista. Denne metoden medfører en viss dødtid, nemlig tiden fra et objekt er mottatt av klienten til neste objekt begynner å komme inn. Dødtiden tilsvarer rundetiden på nettet, regnet fra klienten sender en forespørsel og til responsen kommer tilbake. Det skal ikke så store nettverk til før denne rundetiden gir merkbar treghet. Alternativet til å overføre ett og ett objekt kalles pipelining. Det innebærer at nettleser sender flere forespørsler umiddelbart etter hverandre uten å vente på svar fra webtjener. Dermed skjer overføringen enda litt raskere.

3.2.6.4 Parallelle forbindelser

Dersom nettleseren skal laste ned en lang liste med objekter, kan nettleseren velge å opprette to eller flere parallelle forbindelser mot samme webtjener og begynne å laste ned flere objekter samtidig. Dette er ikke en del av HTTP-protokollen, men en strategi som nettleseren kan velge. De parallelle forbindelsene er hver for seg en HTTP-sesjon. Metoden krever god nettkapasitet. Nettleseren må selvsagt holde styr på hva som kommer inn, og vise alt dette i rett sammenheng.

3.2.6.5 Informasjonskapsler

Formålet med *informasjonskapsler* (*cookies*) er å tilpasse responsen som webtjener gir på visning av en webside du har lest tidligere. Informasjonskapslene lagres på klientens side med informasjon om spesifikke websider. Det er webtjeneren som ber nettleseren opprette disse og bestemmer hvilket innhold som skal lagres. Det gjør webtjener ved å føye til en spesiell headerlinje i responsen:

```
set-cookies: <diverse tilstandsinformasjon>
```

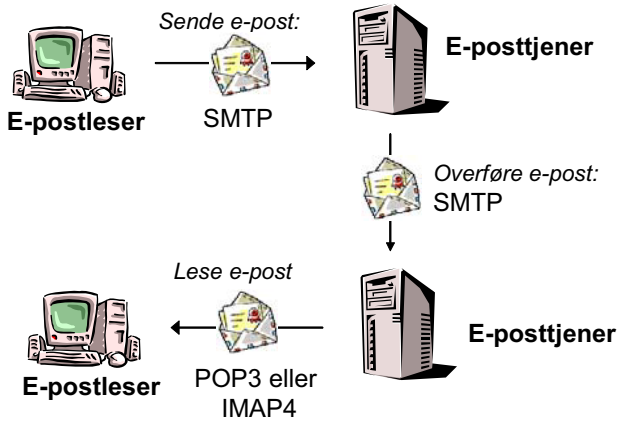
Når nettleseren ved en senere anledning skal spørre etter en webside, sjekkes det samtidig om den har informasjonskapsler som er knyttet til den aktuelle URL-adressen. Hvis så er tilfelle, sendes kapselen sammen med forespørselen. Webtjeneren tar imot og tilpasser svaret best mulig, for eksempel ved å vise ferdig utfylte valgfelter.

Informasjonskapsler kan oppfattes som uønsket lagring av personlig informasjon om hvordan man bruker web, og vi kan konfigurere nettleseren til å være restriktiv med bruken av dem.

3.3 E-post

E-post har eksistert siden Internett ble startet opp. Vi kjenner e-post som et raskt og effektivt hjelpemiddel for kommunikasjon. Man kan sende både til enkeltpersoner og grupper av mottakere og ha elektroniske dokumenter som vedlegg. Hovedelementene i e-postsystemet er klientprogrammet (e-postleseren), e-posttjeneren og protokollene som styrer sendingene mellom disse. Protokollen som styrer sending av e-post, heter SMTP. Den brukes både ved sending fra klientprogrammet og videre mellom e-posttjenere. Det er andre protokoller som styrer klientprogrammets lesing fra e-posttjener, man kan bruke POP3 eller IMAP4.

Formatet på selve e-posten ble først spesifisert i RFC 822 og er siden erstattet av RFC 2822. Formatspesifikasjoner er ikke protokoller; de sier ingen ting om meldingsutvekslingen på nettet, men beskriver hvordan innholdet skal se ut. Dette er et viktig bidrag til å forstå e-post, så vi vil derfor studere dette også. RFC 2822 for e-post tilsvarer HTML for web.



Figur 3.5 Hovedelementer i e-post

3.3.1 E-postklienter

E-postklienter er program som brukes til å sende og motta e-post fra e-posttjenere. Eksempler på populære program er Microsoft Outlook Express, Mozilla Thunderbird og Eudora. Dette er program som installeres på den enkelte PC og må konfigureres med brukernavn og passord mot navngitte e-posttjenere. Alternativt kan man bruke et webgrensesnitt for e-post, hvor det kanskje mest kjente tilbudet

er Hotmail. I dette tilfellet er selve e-postklienten ferdig konfigurert hos tilbyder, man bruker nettleseren og HTTP som et brukergrensesnitt for å få tilgang til klientprogrammet. Fordelen med å skille e-postklient og brukergrensesnitt er at man kan lese e-post fra alle PC-er uten å måtte konfigurere den PC-en man jobber på.

Det å sende og motta e-post er forskjellige oppgaver. I det første tilfellet har man noe man tar initiativet til å få sendt, i det andre tilfellet ber man om å få overført noe som e-posttjeneren har. Oppgavene styres derfor av ulike protokoller, og e-postklienten må håndtere begge oppgavene.

E-postklientene har dessuten diverse funksjoner som hjelper brukeren, for eksempel er adresselister et viktig hjelpemiddel, og filter mot søppelpost er veldig aktuelt. Hvordan dette løses, er ikke standardisert i e-postsystemet og er en konkurransefaktor mellom utviklerne av programvaren.

3.3.2 E-posttjenere

En e-posttjener har to hovedoppgaver: Den skal formidle e-post for sine klienter, og den skal administrere lokale postkasser. Vi tenker oss at Per og hans kollegaer er opprettet som brukere på samme e-posttjener. Når Per sender e-post til en kollega, vil e-posttjeneren se at den er adressert til en lokal bruker. Da blir sendingen plassert direkte i postkassen til vedkommende kollega, som kan lese den ved anledning.

Dersom Per sender e-post til en ekstern kontakt, det vil si med postkasse på en annen e-posttjener, blir e-posten sendt direkte til den eksterne e-posttjeneren. Denne vil i sin tur plassere e-posten i vedkommendes postkasse, klar for avlesing.

Når en e-post er lest, det vil si overført fra e-posttjener til klient, skal da fila slettes på tjenesiden, eller bare markeres som lest? Og skal e-posten kunne flyttes fra innboksen over i andre mapper? Det er to protokoller som brukes for lesing av e-post og styring av filadministrasjonen på e-posttjener: POP3 og IMAP4. De gjennomgås i avsnitt 3.3.5.

En viktig oppgave for de som drifter e-posttjenere, er å sørge for at brukerne unngår søppelpost og virus.

3.3.3 Sending av e-post med SMTP

SMTP (Simple Mail Transfer Protocol) er protokollen for sending av e-post. SMTP ble først definert i RFC 821 og er siden erstattet av RFC 2821. Protokollen brukes både når klientprogrammet sender e-post til sin e-posttjener, og når denne e-posttjeneren «snur seg rundt» og videresender e-posten til mottakers e-posttjener. I det siste tilfellet har e-posttjeneren klientens side av SMTP-forbindelsen. E-posttjenere er derfor eksempel på program som har både en klientside og en tjenerside for én og samme protokoll. Et annet eksempel på slike program er IP-telefoner; man kan både ringe og bli oppringt.

SMTP er kjennetegnet ved at det utveksles kontrollmeldinger (handshaking) før selve e-posten overføres. Formålet med dette er at e-postsystemet kan verifisere at avsender og mottaker har gyldige adresser, det vil si navn som blir akseptert av e-posttjeneren. Når e-posten er overført, har e-posttjeneren også påtatt seg ansvar for videre håndtering av denne, enten den legges i lokal brukerpostkasse eller sendes til en annen e-posttjener. I de tilfellene hvor det ikke lykkes å få e-posten frem til mottaker, blir det sendt et varsel tilbake til avsender om det. Dette er funksjoner som ligger i selve e-postsystemet og utenfor SMTP-protokollen, som kun omhandler sending av e-post.

Her er et eksempel på en SMTP-dialog for å overføre en e-post:

Utlisting 3.2 SMTP-dialog for å overføre e-post

```
 1  220 din.posttjener.no ESMTP
 2  EHLO min.datamaskin.no
 3  250-din.posttjener.no
 4  250-PIPELINING
 5  250 8BITMIME
 6  MAIL FROM: <per@example.com>
 7  250 ok
 8  RCPT TO: <kari@example.com>
 9  250 ok
10  DATA
11  354 go ahead
12  Subject: En liten test
13
14  Her er innholdet i testen.
15  .
16  250 ok 1183717246 qp 15923
17  QUIT
18  221 din.posttjener.no
```

Linje 1: E-posttjeneren presenterer seg når TCP-forbindelsen er etablert. Betegnelsen ESMTP innebærer at e-posttjeneren støtter SMTP-utvidelser (*extensions*).

Linje 2–9: Klienten oppgir navnet sitt, og tjeneren kvitterer med å liste ut SMTP-utvidelsene den støtter. Klienten oppgir videre hvem e-posten er fra, og hvem den skal sendes til. Dette blir kvittert med statuskode 250 hvis alt er funnet i orden.

Linje 10–11: Klienten ber om å få overført data, noe tjeneren gir klarsignal til.

Linje 12–16: Her overføres hele e-posten, inkludert alle vedlegg. Dette til forskjell fra web, der HTTP sender hvert objekt i en egen overføring. Som vi ser, skilles headerfeltene fra meldingskroppen med en tom linje. Avslutning av e-posten markeres med et punktum alene på en linje. SMTP-tjeneren bryr seg ikke noe om innholdet i e-posten, men kvitterer ok med statuskode 250.

Linje 17–18: Klienten sier fra at den ønsker nedkopling, men det kunne eventuelt ha vært overført flere e-poster. Tjeneren kvitterer på at forbindelsen koples ned.

3.3.4 Formatet på e-post

Vi har sett hvordan SMTP styrer utveksling av meldinger for å overføre e-post. Dette er en *protokoll*. SMTP ignorerer innholdet i e-posten, det er bare «data». Det trengs derfor en egen standard som definerer *formatet* på e-post. Formatet er som en mal som bestemmer bruk av tegnsett og informasjonsfelter. Dette formatet ble først definert i RFC 822. Nå er det slik at dette formatet ble bestemt i tidenes morgen i Internett-sammenheng, og det dekker derfor ikke dagens behov. For å slippe å definere et nytt format, noe som ville kreve et enormt arbeid med oppdatering av alle e-posttjenere rundt om i verden, laget man heller en utvidelse, som kalles MIME. Dette blir gjennomgått i det følgende.

3.3.4.1 E-postformat og RFC 2822

RFC 822 definerte det opprinnelige formatet på e-post. Denne standarden omtales som RFC 822, men det egentlige navnet er «Standard for the Format of ARPA Internet Text Messages». Tittelen er interessant nok. For det første har vi betegnelsen *ARPA*, som står for Advanced Research Projects Agency. Dette er den enheten i det amerikanske forsvarsdepartementet som i 1962 startet utviklingen av prinsippene som førte til dagens Internett. Det andre man kan merke seg, er at RFC 822 omhandler *Text Messages*. Man snakket ikke om e-post, det var rett og slett utveksling av enkle tekstmeldinger. Dette forklarer hvorfor man senere trengte MIME for å kunne ivareta nåtidens krav til e-post. RFC 822 er nå erstattet av RFC 2822, som heter «Internet Message Format», men hovedtrekkene er fortsatt de samme.

Formatet definerer en headerblokk og en meldingskropp. I headerblokken finner man diverse linjer med informasjon om sender og mottaker, hvordan e-posten er overført, og hva slags type innhold man får tilsendt. Man kan merke seg at linjene i headerblokken som heter «From» og «To», er helt uavhengige av SMTP-kommandoene «MAIL FROM» og «RCPT TO». Det går derfor an å «lure» mottaker til å tro at e-posten er fra en annen avsender enn det som egentlig er tilfelle, og det er en av problemstillingene knyttet til søppelpost. I meldingskroppen finner man innholdet av e-posten. Meldingskroppen skal være atskilt fra headerblokken av et linjeskift.

RFC 2822 krever at alle tegn i formatet skal være US-ASCII-tegn, som består av tegn med verdier fra 0–127. Det er en ulempe fordi dagens systemer bruker en 8-bit kode for å representere tegn og data, og da kan man ha tegnverdier opp til 255. MIME-standarden gir en løsning på hvordan man konverterer innholdet av e-posten slik at det passer med US-ASCII-formatet.

3.3.4.2 MIME

MIME (Multipurpose Internet Mail Extensions) gir anledning til å bruke nasjonale tegnsett i e-post og sende med filvedlegg av forskjellig format. MIME er definert i RFC 2045 til RFC 2049. Tilpasning til RFC 2822 skjer ved koding av tegnstrøm-

men. Kodingen er en enkel omdanning og må ikke forveksles med kryptering, e-post blir ikke sikret mot innsyn på grunn av dette.

Det er flere kodeformer som kan benyttes, avhengig av hva som sendes, og MIME beskriver hvordan kodingen skal utføres. Når man sender vedlegg i en e-post, er disse atskilt av en grensemarkør, som er en angitt tegnsekvens. E-postleseren viser meldingen pent og pyntelig på skjermen, men innholdet kommer egentlig i en ikke fullt så lesbar form.

Figur 3.6 Slik e-posten vises, og deler av innhold på RFC 2822- og MIME-format

Når man omdanner innholdet, må mottaker varsles om *hvordan* innholdet er omdannet, altså hvilke av MIME-typene som er valgt. Det gjøres ved å sette inn ekstra linjer i headerblokken i e-posten. Dette er for øvrig også samme metode som HTTP bruker for å få ny funksjonalitet inn i en etablert standard, man legger til nye headerlinjer. De to nye MIME-linjene i RFC 2822-headerblokken heter «Content-Type» og «Content-Transfer-Encoding».

Her er et eksempel på omkoding av tekst med nasjonale tegnsett:

```
Content-Type: text/plain;charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable
```

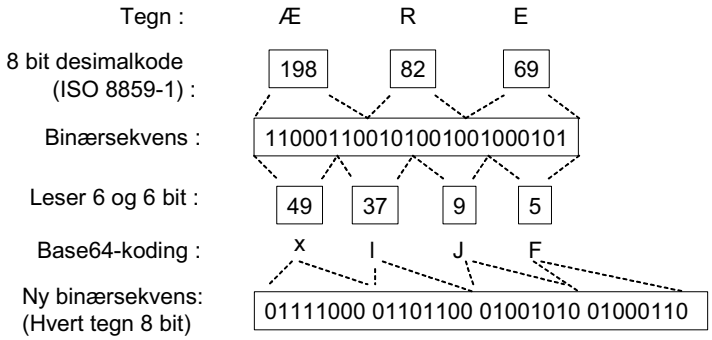
Vi leser i første linje at innholdstypen er vanlig tekst som benytter tegnsettet definert i ISO 8859-1 (Latin 1). Dette tegnsettet inneholder blant annet våre nasjonale tegn æ, ø og å. Det å vite innholdstype gir mottakers e-postleser anledning til å velge en passende visningsform. Er innholdet tekst, skal den vises i tekstvindu; er innholdet en lydfil, kan man starte en avspiller. Neste linje forteller at innholdet er kodet med en metode som kalles «quoted-printable». Denne metoden brukes når tegnstrømmen i hovedsak er bokstaver.

Dersom man sendte med et bilde som vedlegg, ville man få følgende:

```
Content-Type: image/jpeg;name="BILDE.JPG"
Content-Transfer-Encoding: base64
```

Vi leser i dette tilfellet at innholdstypen er et bilde i JPEG-format, og fila heter «bilde.jpg». Bilder oppfattes som binærfiler hvor hvert enkelt byte kan ha tilfeldige verdier fra 0–255. Slikt innhold har man funnet det mer hensiktsmessig å kode om etter en metode som kalles base64.

Base64-koding virker slik at man tar for seg en gruppe på 3 byte (i alt 24 bit) og leser dette som en sekvens på fire tegn, hvert på 6 bit (fortsatt 24 bit). Hvert av disse 6-bit-tegnene blir omdannet til et ASCII-tegn i henhold til en egen Base64-tabell. Dette nye ASCII-tegnet overføres med 8 bit, og dermed øker de opprinnelige 24 bit til 32 bit i overføringen, hvilket er en 33 % økning i datavolum.



Figur 3.7 Base64-omdanning til lovlige ASCII-tegn

En ulempe med MIME er at omkodingen øker datavolumet på det som overføres, samt at det kreves ekstra maskinressurser. Dette oppveies av fordelen ved å slippe å lage en ny standard for å erstatte den etablerte RFC 822. Man kan si at «prisen» for den massive suksessen med e-post er at standarden er blitt fastlåst.

3.3.5 Lesing av e-post med POP3 og IMAP4

Når man mottar e-post, samles de i postboksen (fillageret) på e-posttjener. E-posttjeneren kan sammenlignes med et postkontor som alltid er åpent. Og for å bruke analogien med landpost videre, så har e-post ingen postombæring. Det er brukeren selv som må sjekke om det har kommet e-post. Riktignok kan e-postleseren settes opp til å gjøre dette regelmessig og automatisk, for eksempel med noen minutters mellomrom, men det er like fullt brukerprogrammet som tar initiativet. Da kan man få varsling med lydsignal eller tekst som «You've got Mail». Noen e-posttjenere er også satt opp til å varsle brukeren om ny e-post på SMS.

De to protokollene som brukes for å lese e-post, heter POP3 (Post Office Protocol Version 3) og IMAP4 (Internet Message Access Protocol Version 4). Protokollene omtales også som POP og IMAP i kortversjon. De er definert i henholdsvis RFC 1939 og RFC 2060. Forskjellen mellom disse to ligger i hvordan man kan bruke postkassen. POP3 kom først og har begrenset funksjonalitet, i utgangspunktet er

den laget for å flytte filer over til klienten og slette dem på e-posttjener etterpå. IMAP har mer avansert filadministrasjon og kan lagre filene videre på e-posttjener i ulike mapper. Det har den åpenbare fordelen at e-posten kan leses flere ganger selv om man logger seg på fra forskjellige maskiner.

En forskjell fra sendeprotokollen SMTP er at man må logge seg på med brukernavn og passord for å lese fra postboksen. Man bør være klar over at dersom det ikke er satt opp kryptert pålogging, og det enkleste er å la være, så kan andre på samme nett fange opp brukernavn og passord i klartekst.

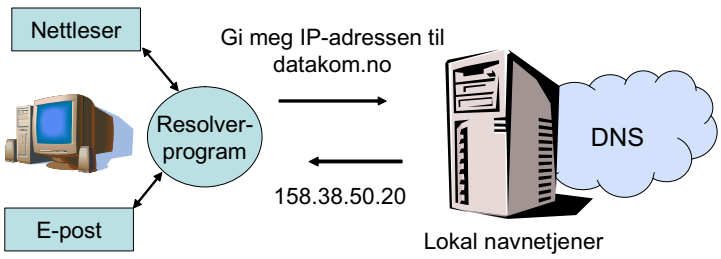
3.4 Navnetjenesten DNS

Navnetjenesten DNS (Domain Name System) brukes for å kople ressursdata til domenenavn. Dette gjør at vi for eksempel kan kople domenenavn med IP-adresser. IP-adressen er den unike tallkombinasjonen som identifiserer hver maskintilkopling på Internett, og vi trenger denne for at datapakker skal finne frem til rett maskin. DNS brukes av alle applikasjoner hvor man adresserer maskiner ved navn, for eksempel i web og e-post. Man kunne ha brukt IP-adresser direkte, men hvem går vel rundt og husker på slike? Telefonnummer kan være vanskelige nok ...

3.4.1 Navneoppslag i DNS sett fra sluttbruker

For å lese websiden som er knyttet til denne læreboka, skriver man i nettleseren *http://datakom.no.* I denne URL-en står domenenavnet *datakom.no*, og nettleseren må ha IP-adressen som skjuler seg bak dette navnet, for å kunne etablere forbindelse til webtjeneren. Så oppgaven er å slå opp i DNS, men hvordan kontakter man DNS? Noen opplysninger må man bli fortalt, og IP-adressen til den lokale navnetjeneren må være lagt inn i datamaskinen på forhånd. Dette konfigureres vanligvis automatisk når strømmen slås på. Konfigurering av PC omtales i kapittel 8.

Vi spør lokal navnetjener: «Hva er IP-adressen til datakom.no?». Og får svar tilbake: «IP-adressen er 158.38.50.20». Dermed vet nettleseren hvilken maskin på Internett den skal etablere forbindelse med.



Figur 3.8 Oppslag i lokal navnetjener

DNS-klienten kalles *resolver* og brukes av alle applikasjoner som trenger DNS-oppslag. Meldingene som blir sendt mellom *resolver*-programmet og lokal navnetjener, kan fanges og analyseres med Wireshark.

3.4.2 DNS og virtuelle webtjenere

En tjenesteleverandør kan tilby ulike virksomheter å ha sine egne hjemmesider på leverandørens maskin. Denne tjenesten kalles *webhotell*. Disse virksomhetene har selvsagt ulike domenenavn, og man får ulike websider vist frem. Man sier at maskinen kjører virtuelle webtjenere for de ulike domenenavnene. Det er header-linjen «*Host:<domenenavn>*» i HTTP som styrer hvilken webside som skal vises, som omtalt i underavsnitt 3.2.5.1.

Poenget her er at det er ingen forutsigbar sammenheng mellom domenenavn og IP-adresse. Forskjellige domenenavn kan kjøres på samme maskin (webhotell), og ett domenenavn kan også kjøres på ulike maskiner (lastdeling). For å vise at samme maskin faktisk kan kjøre to virtuelle webtjenere, kan vi bruke et verktøy for oppslag i DNS. Programmet *nslookup* er preinstallert i de fleste operativsystemer og kan brukes fra kommandolinjen. Nslookup og andre DNS-verktøy omtales i vedlegg A.

Utlisting 3.3 DNS-oppslag med nslookup

```
 1  C:\> nslookup aitel.hist.no
 2  Server: dns.example.com
 3  Address: 192.0.2.10
 4
 5  Non-authoritative answer:
 6  Name:    aitel.hist.no
 7  Address:  158.38.50.20
 8
 9  C:\> nslookup datakom.no
10  Server: dns.example.com
11  Address: 192.0.2.10
12
13  Non-authoritative answer:
14  Name:    datakom.no
15  Address:  158.38.50.20
```

Linje 1: Her sender vi forespørsel om IP-adressen til *aitel.hist.no.*

Linje 2–3: Dette er navnet på den lokale navnetjeneren som svarer, og den har selvsagt også en IP-adresse, som vi finner i linje 3.

Linje 5: Svaret vi får, er ikke-autoritativt, det betyr at lokal navnetjener har mellomlagret informasjonen om det forespurte navnet og dets IP-adresse. Arbeidsdelingen mellom ulike typer navnetjenere omtales i avsnitt 3.4.5.

Linje 7: Her ser vi at IP-adressen til *aitel.hist.no* er 158.38.50.20.

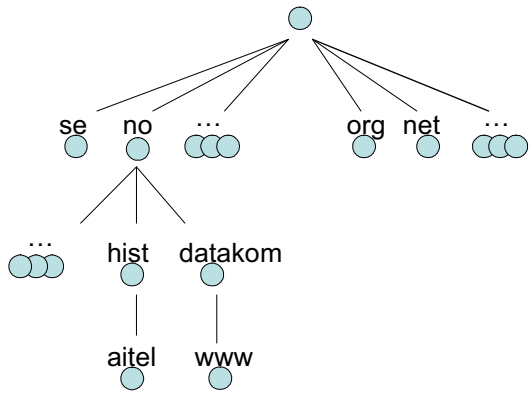
Linje 9: Vi sender en ny forespørsel, nå om IP-adressen til *datakom.no*.

Linje 15: Vi ser at det er akkurat samme IP-adresse for *datakom.no* som for *aitel.hist.no*, nemlig 158.38.50.20.

3.4.3 Strukturen på domenenavn

DNS er hierarkisk strukturert. Et domenenavn består av flere ledd, som kalles domener og er atskilt med punktum. Overordnet ansvar ligger hos ICANN (The Internett Corporation for Assigned Names and Numbers). De bestemmer hvilke toppnivådomener som skal opprettes. I dag er det 13 generelle toppnivådomener, hvor vi finner kjente navn som com, net og org. ICANN avholdt sitt 22. internasjonale møte i april 2005, med over 600 delegater fra 80 nasjoner, og besluttet blant annet å opprette to nye generelle toppnivådomener: jobs og trav. Arbeid og reise er aktuelle tema.

Det er også opprettet flere hundre nasjonale domener, som *no og se*. Under de nasjonale toppnivådomenene kan virksomheter få tildelt domener som arver toppnivådomenet til slutt i domenenavnet. Vi kan illustrere navnetreet slik:



Figur 3.9 Et lite utsnitt av det hierarkisk strukturerte DNS-navnetreet

3.4.4 Forvaltning av domener

Et grunnprinsipp i DNS er delegering av ansvar. Når en virksomhet har fått delegert ansvar for et *domene*, så er det virksomheten selv som bestemmer over opprettelsen av nye underliggende domener. Et slikt underdomene kan rett og slett være «www», slik at det er anledning til å bruke både *http://www.datakom.no* og *http://datakom.no* som likeverdige webadresser.

NORID, som er et datterselskap av UNINETT, har fått delegert ansvar fra ICANN for no-domenet. Oppretting av nye domener under no-domenet gjøres av såkalte registrarer, virksomheter som NORID har gjort avtale med. Registrarfunksjonen kan utføres av virksomheter dersom de oppfyller visse krav. Det er de som har kontakt med kundene, blant annet for fakturering. Det koster noe å etablere og holde et domene, i størrelsesorden noen hundre kroner per år.

NORID har valgt å føre en noe restriktiv navnepolitikk. Det betyr blant annet at det bare er virksomheter (ikke privatpersoner) som har anledning til å registrere domener direkte under no-domenet, og virksomhetene kan bare ha inntil 20 domener hver. Statistikk på NORIDs websider forteller at det ved utgangen av april 2005 var registrert om lag 235 000 domener under no-domenet. Tenk da på alle de andre toppdomenene med sine underdomener, som kommer i tillegg – det er ikke noen liten affære som DNS skal håndtere!

3.4.5 Kort om implementeringen av DNS

DNS er et stort, distribuert system som kjører på hundretusenvis av navnetjenere. Navnetjenere er konfigurerte til å ha ulike roller og funksjoner.

Fra brukerens ståsted møter vi først lokal navnetjener. «Lokal» betyr i denne sammenheng at den er rimelig nær i nettet, slik at responstidene ikke blir så lange hvis pakkene må passere mye utstyr. Lokal navnetjener må ha den egenskapen at den kan «snu seg rundt» og spørre videre på vegne av klient. Denne jobben kalles rekursive oppslag. Lokal navnetjener bør også ha den egenskapen at den kan mellomlagre svar som den mottar (*cache*), slik at den har svaret klart neste gang det gjøres et oppslag på dette domenenavnet. Hvis det ikke har gått for lang tid, da. Det brukes en tidsangivelse for hvor lenge informasjonen skal være gyldig.

På toppen av DNS-navnetreet finner vi rotnavnetjenere. I teorien kunne dette vært en enkelt navnetjener, men på grunn av sikkerhet og trafikkbelastning er denne funksjonen fordelt på flere navnetjenere rundt om i verden. Alle rotnavnetjenerne har en kopi av de samme data. Oppgaven til rotnavnetjenerne er å vite hvilke navnetjenere som er ansvarlige for toppnivådomenene, slik som «.no» og «.org».

Et sted i DNS må det ligge ressursdata som faktisk forteller sammenhengen mellom domenenavn og IP-adresse. For eksempel at webtjeneren for domenenavnet *datakom.no* har IP-adresse 158.38.50.20. Dette er ressursdata som må skrives inn manuelt i navnetjeneren for virksomheten. Navnetjenere som har «fasiten» på sammenheng mellom domenenavn og IP-adresse, kalles autoritative navnetjenere. Navnetjenere som mellomlagrer slik informasjon, kalles ikke-autoritative navnetjenere.

Vi kan lagre ulike ressurstyper i DNS. I tabellen ser vi noen av de vanligste typene.

Type	Betyr	Inneholder	Eksempel
A	Address	IP-adresse	10.0.0.50
NS	NameServer	Navnetjener for et domene	ns.example.com
CNAME	Canonical Name	Alias til et annet domenenavn	pc1.example.com
PTR	Pointer	Domenenavnet for reversoppslag på IP-adresse	pc1.example.com
MX	Mail eXchanger	Navn og prioritet på SMTP-tjenerne som behandler post for domenet	10 mail.example.com

For eksempel kan e-posttjener og webtjener ha samme domenenavn, men ulike IP-adresser. Disse skilles ved at man spør etter ressurstyper av typen «A» eller «MX». Man kan også ha flere alias for én og samme maskin.

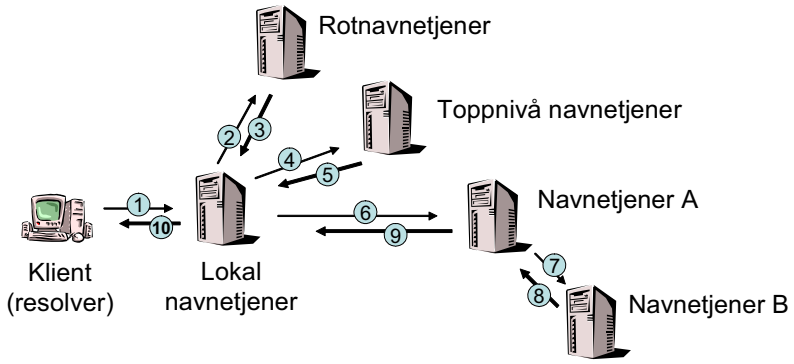
3.4.6 DNS-håndtering av oppslag fra lokal navnetjener

Vanligvis sender brukeren forespørsler til en lokal navnetjener som kan mellomlagre svar. Men såfremt svaret ikke allerede er mellomlagret, må den spørre videre. Det er bare ett sted å henvende seg om helt ukjente toppnivådomener: Vår lokale navnetjener spør en rotnavnetjener «Hva er IP-adressen til dette ukjent domenenavnet?». Rotnavnetjenere holder ikke styr på slike detaljer, men de kan hjelpe ett skritt videre, de vet hvem som er toppnivå navnetjenere for alle domenenavn, og svarer tilbake.

Lokal navnetjener har nå fått et svar, men ingen løsning, så den må spørre videre. Denne gangen er det toppnivå navnetjener for det etterlyste domenenavnet som spørres. Heller ikke her kan vi forvente å få vite IP-adressen, til hvert toppnivå domenenavn er det svært mange underdomener å holde oversikt over. Igjen får vi svar tilbake om hvor vi kan spørre videre. Til slutt har vi vandret langs greinene i navnetreet helt til kilden, den autoritative navnetjeneren, gitt at det ikke har dukket opp mellomlagret informasjon underveis.

Dette eksemplet beskriver et rekursivt navneoppslag. «Rekursivt» betyr at noe går i flere ledd, og innebærer at en navnetjener spør videre på vegne av en annen helt til den finner et svar. Lokal navnetjener er satt opp til å jobbe rekursivt på vegne av klienten. «Iterativt», derimot, betyr at noe gjentas, og navnetjenere som ikke ønsker

å ta ansvaret med å jobbe rekursivt, svarer bare med en henvisning til hvor det kan spørres videre. Man kan, på grunn av de mange weboppslag på nettet, forstå at rotnavnetjenere ikke er satt opp til å jobbe rekursivt.



Figur 3.10 Navneoppslag i DNS, hvor lokal navnetjener og navnetjener A foretar rekursive forespørsler

Lokale navnetjenere er avhengige av at de på forhånd vet hvor rotnavnetjenerne finnes, det vil si kjenner deres IP-adresser. Disse legges inn ved oppstart av navnetjeneren, og det kan man gjøre fordi alle rotnavnetjenere har faste IP-adresser. Samme metode brukes når en PC får IP-adressen til lokal navnetjener.

3.4.7 Flere funksjoner i DNS

Hovedhensikten med DNS er å gi oss IP-adresser når vi slår opp på domenenavn. For at DNS skal fungere optimalt, har vi også mulighet for lastdeling på webtjenere, dynamisk oppdatering av DNS-poster, bruk av alias-domenenavn og reversoppslag.

Lastdeling trengs når populære websider har så mange oppslag at det må brukes flere maskiner med nøyaktig samme innhold. Et domenenavn kan gi flere IP-adresser til svar, og for hver gang det spørres, blir rekkefølgen av disse rotert ett hakk. Dermed blir lasten, det vil si forespørsler fra mange nettlesere, fordelt blant maskinene.

Dynamisk DNS er kjekt å ha dersom du skulle ha en tjener med IP-adresse som kan endres. Dette er typisk tilfelle i private nett med dynamisk tildelt IP-adresse fra ISP. En konsekvens av dette er at DNS kan gi svar som ikke lenger stemmer, slik at ingen lenger finner tjeneren. Løsningen er at DNS-protokollen takler pålogging med brukernavn og passord og automatisk oppdatering, altså at det legges inn ny IP-adresse i ressursposten i DNS. Dette er en rutine som kjøres hver gang det har vært en restart i sluttbrukerutstyret.

Revers DNS-oppslag er en forespørsel til DNS på IP-adresse for å få domenenavn i retur, altså det motsatte av hva systemet vanligvis brukes til. Denne funksjonen er nyttig hvis man skal endre en nettverkstilkopling og trenger å vite hvilke domenenavn som peker til denne IP-adressen. Funksjonen kan også brukes i forbindelse med kontroll av avsendere av e-post.

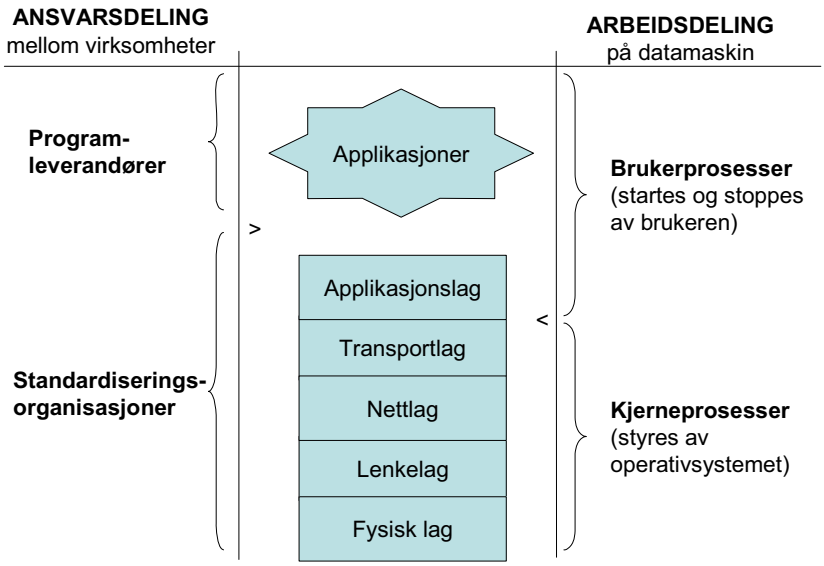
3.5 Om applikasjonslagets protokoller

Vi har nå studert de tre mest brukte anvendelsene på Internett: web, e-post og navnetjenesten. De har samme hovedstruktur, nemlig protokoller som sender meldinger mellom klient og tjener. Disse protokollene ligger på *applikasjonslaget* i lagmodellen.

Applikasjonslagets oppgave er å tilby protokoller for utveksling av meldinger mellom klient- og tjenerprogram. Typiske protokoller: HTTP, DNS, SMTP.

Selve applikasjonen (programmet) regnes ikke med til applikasjonslaget. Applikasjonen er noe som ulike produsenter utvikler i konkurranse med hverandre, og det er deres ansvar hvordan de virker. Det er ingen krav til hvordan applikasjonen skal se ut. Det som derimot er et krav til alle applikasjoner, er at de må følge protokollene for tjenesten. Nettlesere må følge protokollen for meldingsutveksling på web, nemlig HTTP. E-posttjenere må kunne sende og motta e-post med SMTP. Det er de standardiserte protokollene som gjør at programvare fra ulike produsenter kan fungere side om side og i samspill med hverandre. De to hovedorganisasjonene som har ansvar for å standardisere applikasjonslagets protokoller, er W3C og IETF.

Applikasjoner og tilhørende protokoller har et fellestrekk: De er brukerprosesser som går på maskinen så lenge applikasjonene kjører (det vil si bruker maskinressurser). Det er forskjellig fra den underliggende protokollstakken for kommunikasjon, som er en del av operativsystemets kjerneprosesser. Disse kjøres hele tiden, uavhengig av om det er noen aktivitet på nettet eller ikke.



Figur 3.11 Sammenhengen mellom applikasjoner og lagdelt kommunikasjonsmodell

Fakta, diskusjonstema og nyttige lenker

Fakta om web

- Web er en klient–tjener-applikasjon hvor nettleseren sender en forespørsel og webtjeneren gir svar.
- Webprotokollen heter HTTP (HyperText Transfer Protocol) og er en av de mest brukte protokollene på Internett.
- HTTP er definert i RFC 1945 (versjon 1.0) og RFC 2616 (versjon 1.1).
- En webside består ofte av en hovedtekst og flere objekter, objektene lastes ned med separate HTTP-forespørsler.
- Nettleseren kan bruke objekter fra lokalt mellomlager dersom de ikke er endret på webtjeneren siden sist, HTTP kan teste på dette.
- Man kan ha vedvarende oppkopling og pipelining i HTTP v 1.1.

Fakta om e-post

- E-posttjenere har to oppgaver: formidling og oppbevaring av e-post.
- Sending av e-post styres av SMTP (Simple Mail Transfer Protocol).
- Lesing av e-post styres av POP3 (Post Office Protocol, versjon 3) eller IMAP4 (Internet Message Access Protocol, versjon 4).
- All e-post må ha et bestemt format. Formatet er definert i RFC 2822.
- Nasjonale tegnsett og filvedlegg må kodes om for å passe til dette formatet, og denne tilpasningen er gjort med MIME-standarden.

Fakta om DNS

- DNS (Domain Name System)-konsepter er spesifisert i RFC 1034 og implementeringer i RFC 1035. Det er en rekke senere oppdateringer til disse.
- DNS oversetter mellom domenenavn og IP-adresser.
- DNS bygger på delegering, både administrativt (forvaltning av domenenavn) og teknisk (distribuerte navnetjenere).
- Det er ingen forutsigbar sammenheng mellom domenenavn og hvilke IP-adresser de er assosiert med.

Diskusjonstema

- Legger man igjen elektroniske spor når man bruker web?
- Er det trygt å handle i nettbutikker?
- Hvordan kan man unngå uønsket e-post (spam, søppelpost)?
- Kan man bruke nasjonale tegnsett i domenenavn?

Nyttige lenker

IETF har en arbeidsgruppe som undersøker hvordan man kan kryptere vedlegg i e-post med en ny S/MIME-standard. Her er en lenke som viser hva de jobber med:
http://www.ietf.org/html.charters/smime-charter.html

ICANN, ansvarlig for administrasjon av navn og nummer på Internett, gir oversikt over hvilke toppnivådomener som finnes, og lenker til de som forvalter disse:
http://www.icann.org/

NORID, de som forvalter no-domenet, gir retningslinjer for registreringsregler, tips og veiledning, oversikt over registrarer og registreringsstatistikk:
http://www.norid.no/

Her er en hjelp for de som ønsker å teste egen DNS-konfigurering:
http://www.rscott.org/dns/