TK1100
FORELESNING 0x08 TRANSPORTLAGET

1. Transportlaget

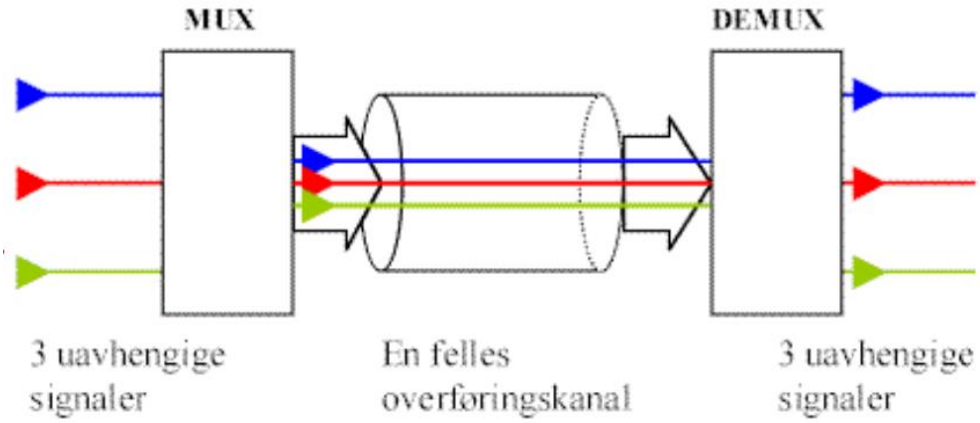
a) Kort forklart, hva er hensikten med transportlaget?

Håndtering av transport for applikasjonsmeldinger mellom klient og tjener til en applikasjon, overføringsenheten er et segment. Det er altså protokollen i transportlaget som definerer forbindelser til de individuelle portene, når en skal sende/levere en pakke. Protokollene fungerer på toppen av IP-protokollene. IP-protokollene er i nettverkslaget og dette er tema for neste forelesning.

b) Hvilke typer protokoller brukes i transportlaget?

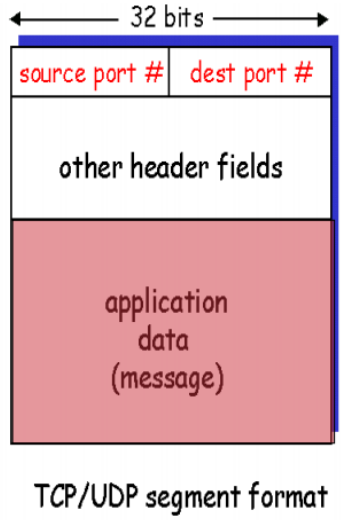
TCP og UDP.

c) Forklar med tegning hvordan multipleksing og demultipleksing fungerer.



d) Hva er et portnummer, og hvilken fordel gir det oss når vi sender data fra en maskin til en annen?

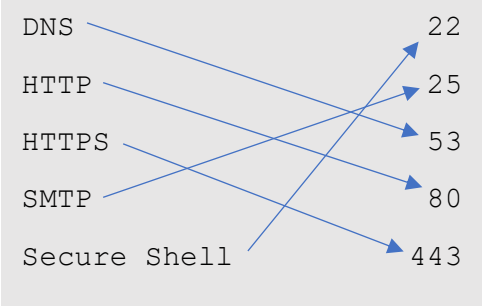
Segmentet til TCP eller UDP protokoller inneholder portnummer til sender og mottaker (som vist på bildet under).
Et portnummer er som en type ID, som brukes for at maskinen som får tilsendt data, skal vite hvilken applikasjon som skal motta den. Prosessene på maskinen må altså si hvilken port de vil forvente svar fra, *dest port.* Man kan også ved hjelp av *source port*, finne ut av hvor på senderens maskin dataen ble sendt fra.
Portnummeret hjelper oss altså å vite hvor på *maskinene* dataen blir sendt til og fra.



e) Hva menes med en «trafikk-kork»?

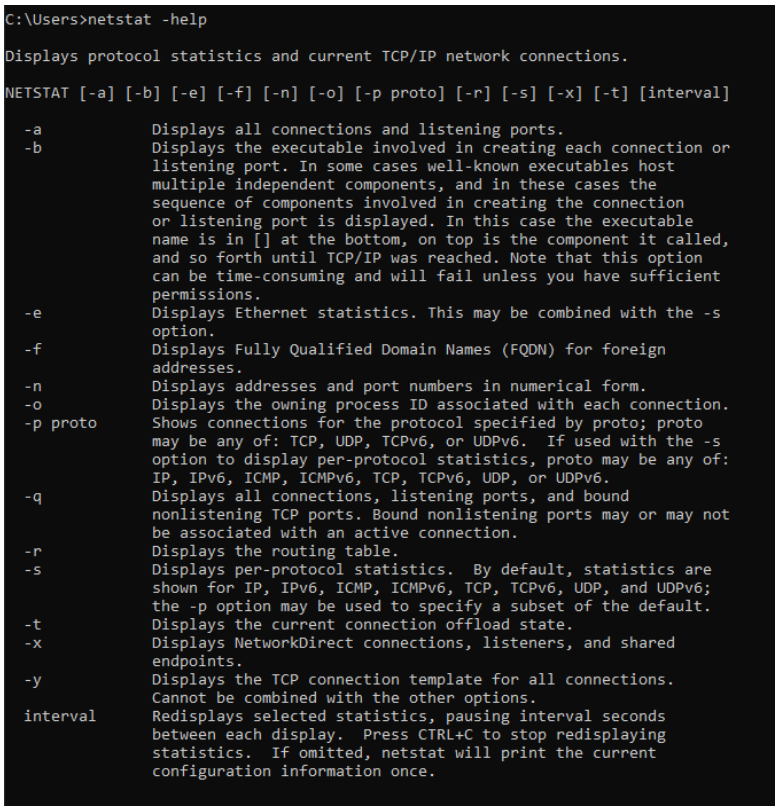
Dette skjer når det er for mange kilder som sender for mye data for fort til at de som skal transportere dataen ikke klarer å håndtere det. Disse trafikk-korkene skjer oftest i routerne som skaper nettverket dataen skal transporteres på. Når en trafikk-kork oppstår kan dette resultere i at pakker blir tapte eller at det blir lange forsinkelser fordi det er kø.

f) Sett samen protokoll med riktig port.



g) Hvilken type informasjon får du frem med å kjøre disse kommandoene i terminalen:

- `netstat -help`
  `(Se bilde under)`



- `netstat -a`
  `Displays all connections and listening ports.`

- `netstat -n`
  `Displays addresses and port numbers in numerical form`

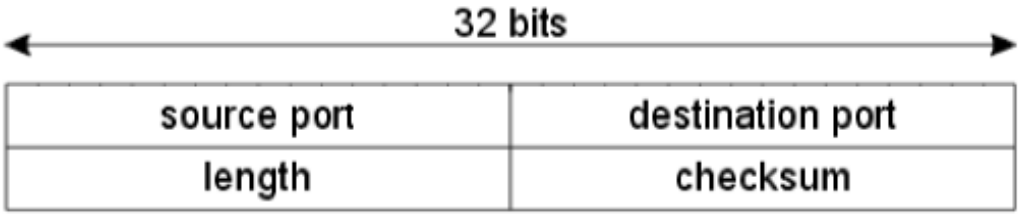
2. UDP

a) Skriv **150-250** ord om UDP, hva er det? Hva brukes det til? Hva er bra/dårlig med UDP? Hvordan oppdager man feil i UDP protokollen?

Punkter du burde ha vært innom:

- Tap av segmenter
- Rekkefølge av segmenter
- Handshake/etablering av forbindelse
- Ingen forsinkelse på grunn av ingen forbindelse
- Segment headern er liten
- Ingen kontroll av trafikk-kork
- Protokollen gir mulighet for bruk av broadcasting
- Feil-håndtering kan tas hånd om av mottakerens applikasjon
- UDP sin sjekksum og hvordan den fungerer

b) Fyll i UDP headern



c) Om vi regner på en UDP sjekksum, ville da disse pakkene inneholde feil? Hint: se slideserie for å finne ut av hvordan du skal regne dette, husk «wrap around»

Pakke 1 – utregning

Addere de første 16-bit dataene

```
  1010 0110 1010 1101
+ 1101 0010 0011 1010
1
  0111 1000 1110 0111
```

Wrap around gir svaret:
0111 1000 1110 1000

Ta svaret fra forrige og addere med neste 16-bit data

```
  0111 1000 1110 1000
+ 1000 1011 0100 1001
1
  0000 0100 0011 0001
```

Wrap around gir svaret:
0000 0100 0011 0010

Ta svaret fra forrige og sjekk det mot sjekksummen

```
  0000 0100 0011 0010
+ 1101 1011 1100 1011
  1101 1111 1111 1101
```

Svaret ovenfor er ikke kun 1-ere og derfor inneholder denne pakken en feil

Pakke 2 – utregning

Addere de første 16-bit dataene

```
  1010 1110 1010 1001
+ 0010 1011 1001 0101
  1101 1010 0011 1110
```

Ta svaret fra forrige og addere med neste 16-bit data

```
  1101 1010 0011 1110
+ 1100 1010 0110 0101
1
  1010 0100 1010 0011
```

Wrap around gir svaret:
1010 0100 1010 0100

Ta svaret fra forrige og sjekk det mot sjekksummen

```
  1010 0100 1010 0100
+ 0101 1011 0101 1011
  1111 1111 1111 1111
```

Svaret ovenfor er kun 1-ere og derfor er denne pakke riktig

3. TCP

a) Hvorfor brukes det mer TCP enn UDP i dag?

Med UDP som protokoll kan segmenter gå tapt eller leveres i feil rekkefølge. Det utføres heller ingen sikker forbindelse mellom avsender og mottaker gjennom for eksempel en handshake. Med andre ord er TCP generelt mer sikkert med overføringen av dataen og du kan være sikker på at all data kommer frem til mottaker. TCP er også en protokoll som gir mulighet for at både sender og mottaker kan sende data samtidig. Dette kalles for «full duplex».

Likevel må det nevnes at UDP har veldig rask overføring, og dermed egner seg bra til applikasjoner som vil være raske, ikke skades ved å miste litt data. Eks: Enkelte spill og videostreaming

b) På hvilken måte blir TCP pålitelig?

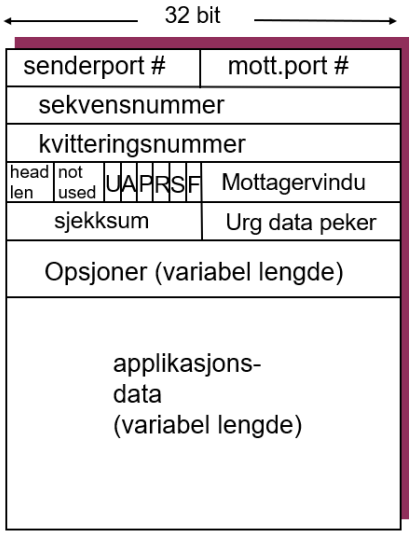
Fordi TCP bruker seg av punkt-til-punkt transformering av data. Her er det altså kun en avsender og en mottaker. For at avsender og mottaker skal kunne overføre data mellom hverandre, etablerer de en kobling gjennom en «handshake». Denne handshaken gjør at de får en sikker oppkobling helt til overføringen av dataen er ferdig. Men kanskje det viktigste med TCPs pålitelighet, er at til forskjell fra UDP, har TCP feil-håndtering, slik at applikasjonen selv slipper å ta hånd om dette.

c) Hvordan håndterer TCP problem som oppstår?

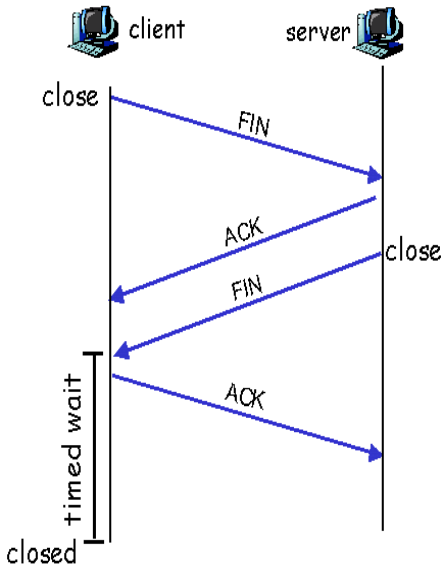
Ved hjelp av sjekksum, kvittering, sekvensnummer og timer. Under «For valgfritt egenstudier» i forelesningen er det beskrevet i detalj hvordan TCP bruker disse for å håndtere feilen som kan oppstå ved transport av data.

d) Er TCP sin header større eller mindre enn UDP sin header?

TCP sin header er større, for å få plass til alt av ekstra data for feilhåndtering. For eksempel har TCP headern et 32-bit kvitteringsnummer som UDP ikke har.



e) Tegn opp en nedkobling av en TCP-forbindelse.



f) Hva menes med TCP og rettferdighet?

Om du har to maskiner på samme nettverk og de skal bruke samme router for å transportere data ved hjelp av TCP, er det meningen at dataraten skal bli rettferdig. Med andre ord, skal de begge få lov til å bruke ressursene like mye. Det finnes forskjellige måter å regne denne rettferdigheten på, men det vanligste er å si: datarate / antallet TCP-sesjoner **eller** R/K